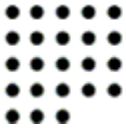

Diplomarbeit

„Konzeption und Realisierung eines Statistiktools auf Basis von JSP und DOJO“

vorgelegt an der

Fachhochschule Köln, Campus Gummersbach

Fakultät für Informatik und Ingenieurwissenschaften

im Studiengang Allgemeine Informatik

Ausgearbeitet von:

Eduardo Wildt Graziani Matr.-Nr.: 11030709

Erstprüfer: Prof. Dr. Heide Faeskorn-Woyke

Zweitprüfer: Prof. Dr. Birgit Bertelsmeier

Gummersbach, im August 2013

Inhaltsverzeichnis

1. Einleitung

Das Institut für Informatik der Fachhochschule Köln betreibt ein e-Learning-Datenbank-Portal (EDB-Portal) für das Studienfach „Datenbanken I und II“. In diesem Portal werden den Studenten verschiedene e-Learning -Anwendungen als Trainer zur Verfügung gestellt, die das Lernen über das Thema Datenbank und auch die Vorbereitung auf die Klausuren unterstützen sollen. Jeder Trainer dient der Übung eines bestimmten Themengebiets, das in der Vorlesung vorkommt. Alle Trainer auf dem EDB-Portal wurden in Java Server Pages (JSP) implementiert.

Für ein solches Portal ist es wichtig zu beobachten, wie jeder Trainer von den Studenten verwendet wird und auch, ob die Studenten die Aufgaben richtig lösen können. Solche Informationen werden erfasst, nachdem ein Projekt von einer Studenten-Gruppe realisiert wurde, das alle Trainer des EDB-Portals überwacht und entsprechende Informationen sammelt. Dafür wurde eine Java-Klasse mit Methoden erzeugt und in das EDB-Portal integriert. Bei bestimmten Aktionen der Benutzer auf dem EDB-Portal werden die Methoden automatisch aufgerufen. Diese Methoden speichern Daten der Aktionen in das für das Projekt erstellte Statistiktool-Schema.

Die Diplomarbeit „Konzeption und Realisierung eines Statistiktools auf Basis von JSP und DOJO“ befasst sich mit der Entwicklung eines Statistiktools, das die gesammelten Daten aus dem Statistiktool-Schema auf einer Benutzeroberfläche einfach und übersichtlich darstellen soll. Das Ziel des Statistiktools ist, den EDB-Administratoren und Professoren Informationen über die Verwendung des EDB-Portals zu liefern.

Das Statistiktool wurde, wie die anderen Anwendungen auf dem EDB-Portal, als JSP-Anwendung implementiert. Wegen der großen Menge an gesammelten Daten ist eine übersichtliche Darstellung der Daten nur als Diagramme möglich. Zur Erstellung von Diagrammen musste ein Framework im Statistiktool eingesetzt werden. Aus diesem Grund wird das Dojo Toolkit im Statistiktool verwendet, für das erst nach einer Analyse und Bewertung mit der Entwicklungsumgebung Oracle Application Express (APEX) entschieden wurde. Aus dem Vergleichen zwischen Dojo Toolkit und APEX bildet sich ein Schwerpunkt der Arbeit,

der im Kapitel „3.4.5 Vergleich zwischen APEX und DOJO“ beschrieben wird. In diesem Kapitel wird begründet, warum das Dojo Toolkit für das Statistiktool ausgewählt wurde.

Mit den Informationen aus den Diagrammen sollen Erkenntnisse gewonnen werden, die später in die Vorlesung „Datenbanken I und II“ einfließen können. Diese Informationen sollen zeigen, in welchen Themen beispielsweise die Studenten am meisten Fehler machen, so können diese Themen in der Vorlesung nochmal behandelt werden. Oder wenn beispielsweise einige Trainer nicht so oft aufgerufen werden, dann können die Professoren die Studenten motivieren, diese Trainer zu verwenden und dadurch deren Themen zu lernen.

Ein anderer wichtiger Schwerpunkt der Arbeit ist, die gesammelten Daten zusammen zu berechnen und in den Statistik-Tabellen zu speichern. Damit konnte ein Performance-Problem im Statistiktool vermieden werden, das bei der Analyse des Statistiktool-Schemas aufgedeckt wurde.

Diese Diplomarbeit besteht aus einem theoretischen und einem praktischen Teil. Der theoretische Teil unterteilt sich in zwei Abschnitte. Der erste Abschnitt befindet sich im Kapitel 2, in dem die Grundlagen von JSP und Dojo erläutert werden, die zum besseren Verständnis dieser Arbeit benötigt werden. Der zweite Abschnitt steht im Kapitel 3 und dort wird die Konzeption zum Statistiktool beschrieben. Nach der Festlegung von Anforderungen und Vorgaben an das Statistiktool kommt ein wichtiger Teil dieses Kapitels, das sich mit der Analyse des Statistiktool-Schemas und deren Erkenntnisse beschäftigt. Erst mit den gewonnenen Erkenntnissen konnte festgelegt werden, welche Funktionen das Statistiktool anbieten kann und auch, welche Informationen es liefern kann. Zu einem anderen wichtigen Teil des Kapitels zählen auch den Vergleich zwischen APEX und Dojo.

Im Kapitel 4 wird die Realisierung des Statistiktools erläutert, was den praktischen Teil der Arbeit darstellt. Der Schwerpunkt des Kapitels liegt bei der Implementierung des Statistiktools und die Erweiterung des Statistiktool-Schemas durch die Statistik-Tabellen.

Das Kapitel 5 schließt diese Arbeit ab, in dem die Erkenntnisse bewertet und reflektiert werden, die im Rahmen dieser Arbeit entstanden sind.

2. Grundlagen

2.1 Web-Anwendungen mit Java Server Pages

Web-Anwendungen mit Java Server Pages (JSP) basieren auf der Java Plattform-Enterprise Edition-Technologie (Java-EE), die von Sun Microsystems entwickelt wurde. Java-EE ist eine Softwarearchitektur-Spezifikation für die Entwicklung von serverseitigen Web-Anwendungen und auch anderen Anwendungen, die in Java programmiert sind. Mit dieser Technologie können dynamisch generierte Web-Seiten erzeugt werden.

Bei einer Web-Anwendung mit der Java-EE-Technologie entstehen folgende Vorteile:

- gute Performance
- einfach anpassbare und austauschbare Komponenten
- Nur serverseitige Anwendung
- Portabilität auf andere Systeme
- die Möglichkeit, die Anwendung einfach zu erweitern
- einfache Wartung

In den folgenden Kapiteln werden die wichtigen Komponenten vom Java-EE beschrieben.

2.1.1 Java-EE-Server

Ein Java-EE-Server ist eine Server-Anwendung, die der Implementierung von der Java EE-Spezifikation entspricht und die Standard-Java-EE-Dienste anbietet. Ein Java-EE-Server besteht aus Komponenten, für die Dienste in Form von Container zur Verfügung gestellt werden. Diese Container bilden eine Schnittstelle zwischen den Komponenten und den untergeordneten Funktionen von der Plattform, die diese Komponenten unterstützen.

Es gibt drei Arten von Containern in einem Java-EE-Server: Anwendung-Client-Container, Enterprise-JavaBeans-Container und Web-Container. Durch die Anwendung-Client-Container können Java-Anwendungen, die auf dem Client-Rechner ausgeführt werden, mit dem Java-

EE-Server kommunizieren, indem diese Anwendung Komponenten auf dem Java-EE-Server verwenden. Der Enterprise-JavaBeans-Container (EJB-Container) ist für die Anwendungslogik auf dem Java-EE-Server zuständig und verwaltet die Enterprise JavaBeans. Der Web-Container dient der Darstellung der Web-Seiten, die durch Servlets und Java Server Pages realisiert wird. Der Web-Container kann aber auch auf den EJB-Container zugreifen, um Daten für die Web-Seiten zu erhalten. Der Web-Container enthält einen Servlet-Container (z. B. Tomcat), der auch als Servlet-Engine bezeichnet wird und für die Ausführung und Verwaltung von Servlets verantwortlich ist.

Man kann sich unterschiedliche Anwendungsszenarien für das Java-EE-Programmiermodell vorstellen, denn nicht alle Java-EE-Server haben einen Web-Container und einen EJB-Container, weil nicht jede Anwendung beide Container benötigt.

Die folgende Abbildung zeigt alle Java-EE-Container und ihre Beziehungen.

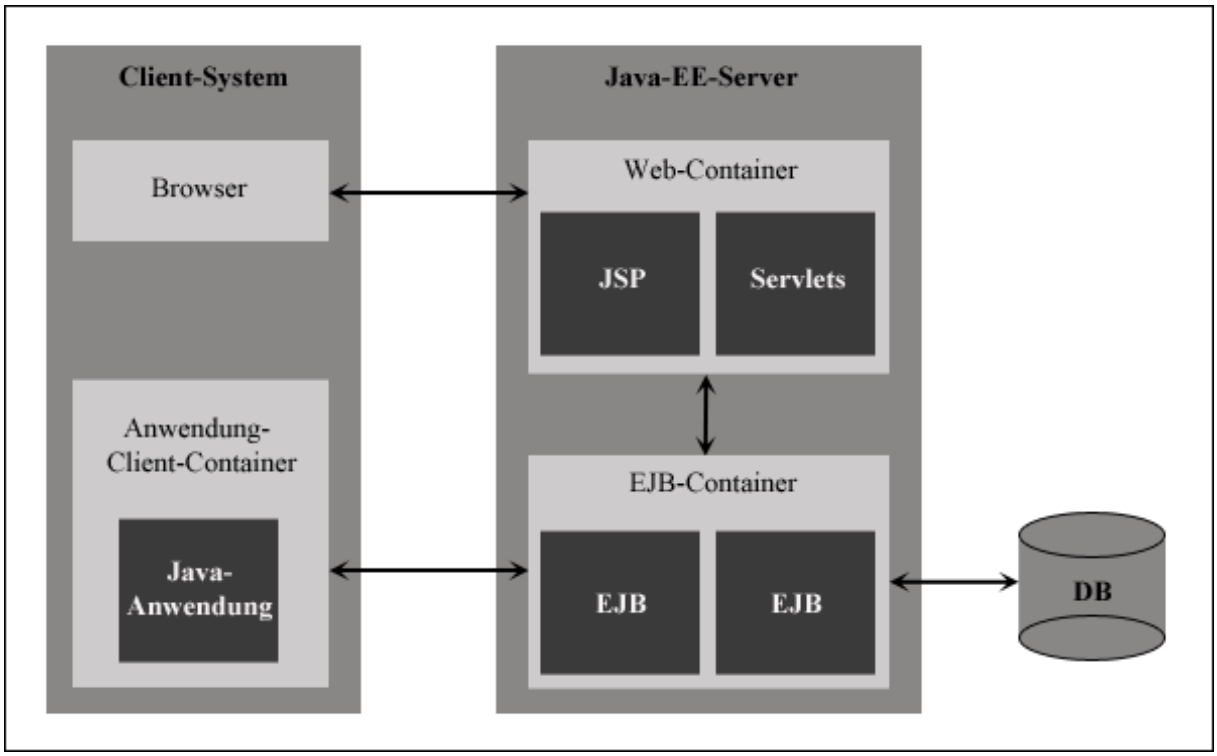

Abbildung 1: Java-EE-Container [1]

[1] *Vgl. Evans (2013)*

Die Anwendungsarchitektur von Web-Anwendungen mit Java Server Pages kann man durch die Abbildung 1 „Java-EE-Container“ gut erkennen. Eine Anwendungsarchitektur besteht immer aus einer Präsentationsebene, einer Anwendungsebene und einer Datenebene. Die Präsentationsebene wird durch die von JSP und Servlets erzeugten Web-Seiten dargestellt. In der Anwendungsebene sind die Anwendungsfunktionen, die auf die Daten der Anwendung ausgeführt werden. Diese Funktionen werden in Servlets und EJB realisiert. In der Datenebene werden die Daten verwaltet, die die Web-Anwendung benötigt. Diese Aufgabe kann von einem Datenbank-Server umgesetzt werden. Die Daten der Web-Anwendung müssen sich aber nicht immer auf einem Datenbank-Server befinden. Sie können auch auf dem Server, z. B. in XML-Dateien, gespeichert werden.

2.1.2 Java Server Pages

Java Server Pages (JSP) sind Web-Seiten, die aus einem HTML-Dokument mit eingebettetem Java-Code bestehen. Im Gegensatz zum HTML-Code dient der Java-Code nicht der Ausgabe auf der Web-Seite.

In einer JSP-Datei wird der Java-Code immer zwischen den Tags <% und %> platziert, mit denen man den Java-Code vom HTML-Code in einer JSP-Datei deutlich unterscheiden kann. Dieser Unterschied stellt den großen Vorteil bei der Entwicklung von Java Server Pages dar, weil eine Trennung zwischen Darstellung und Anwendungslogik einer Web-Anwendung vorgenommen werden kann. So kann das Layout einer Web-Seite gestaltet werden, ohne dass die Web-Entwickler Java-Programmierkenntnisse besitzen.

Wenn ein Browser eine JSP-Datei anfordert, ruft der Webserver den Servlet-Container auf. Der Servlet-Container prüft zuerst, ob ein entsprechendes Servlet für die JSP-Datei bereits existiert. Falls es noch kein Servlet gibt, führt der Servlet-Container eine Umwandlung der JSP-Datei zu einem Servlet durch. Das generierte Servlet gibt dann den HTML-Code der JSP-Datei aus und führt den zugehörigen eingebetteten Java-Code aus. Nach der Übersetzung des Codes wird das Servlet vom Servlet-Container kompiliert, geladen und ausgeführt. Aus diesem Vorgang wird eine Webseite generiert, die an den Browser zurückgesendet wird.

Java Server Pages werden auch häufig mit Servlets verwendet, indem JSP-Dateien mit Hilfe von Formularen Anfragen an Servlets absenden. Die Servlets erhalten dann die Daten aus dem Formular, verarbeiten sie und erstellen eine passende Antwort.

Die folgende Abbildung zeigt ein Beispiel von einer JSP-Datei mit Java-Code und einem HTML-Formular. Im Java-Code wird ein Objekt von Typ Calendar erzeugt. Das Objekt enthält das aktuelle Datum, aus dem die Uhrzeit mit der Methode GET zurückgegeben wird. Die Uhrzeit wird in der IF-Anweisung verwendet, um den richtigen Text im HTML-Code auszugeben. Der HTML-Code enthält ein Formular, in das ein Name und ein Nachname eingegeben werden sollen. Beim Betätigen des OK-Buttons werden diese Daten an das Servlet „HalloServlet“ gesendet, das sie bearbeiten soll. Im Kapitel „2.1.3 Servlets“ wird dieses Beispiel weiter verfolgt und dort wird es erklärt, wie die abgesendeten Daten dieses Formulars vom Servlet bearbeitet werden.

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
<%
        java.util.Calendar aktStunde = new java.util.GregorianCalendar();
        String begruessung;

        if(aktStunde.get(aktStunde.HOUR_OF_DAY) < 12)
                begruessung = "Guten Morgen!";
        else if(aktStunde.get(aktStunde.HOUR_OF_DAY) < 18)
                begruessung = "Guten Tag!";
        else
                begruessung = "Guten Abend!";
%>

<h1> <%= begruessung %></h1>
<br />
<form method="post" action="./HalloServlet">
        Name <input type="text" name="name">
        <br />
        Vorname <input type="text" name="vorname">
        <br />
        <input type="submit" name="button" value="OK">
</form>

</body>
</html>
```

Abbildung 2: Beispiel von einer JSP-Datei mit Java-Code und einem Formular

2.1.3 Servlets

Servlets sind Java-Klassen, die auf einem Web-Server ausgeführt werden und Anfragen von Clients über das HTTP-Protokoll annehmen, verarbeiten und beantworten können. Servlets dienen den Anwendungslogikaufgaben einer Webanwendung. Zu Beispielen von Anwendungslogikaufgaben gehören das Laden von Daten aus einer Datenbank und die Ausführung

von Berechnungen. Servlets können aber auch die Antworten auf die Anfragen dynamisch generieren und an die Clients senden.
Alle Servlet-Klassen werden kompiliert und in eine Web-Archiv-Datei (WAR-Datei) umgewandelt, die dem Servlet-Container übergeben wird. Beim Aufruf von einem Servlet leitet der Web-Server die HTTP-Anfrage an den Servlet-Container weiter, der eine Instanz des Servlets erstellt. Dazu erzeugt der Servlet-Container ein Objekt von Typ HttpServletRequest mit Daten der Anfrage und ein Objekt von Typ HttpServletResponse zur Generierung einer Antwort. Danach ruft der Servlet-Container die Methoden des Servlets auf, die die Anfrage bearbeiten soll. Welche Methode aufgerufen wird, wird aus dem HTTP-Header der Anfrage entnommen.

Ein Servlet kann auf dem Web-Server nur aufgerufen werden, wenn bestimmte Metainformationen zur Verfügung stehen. Diese Metainformationen beschreiben die Web-Anwendung und werden in eine XML-Datei (web.xml) der Web-Archiv-Datei gespeichert, die Deployment Descriptor genannt wird. Der Deployment Descriptor enthält z. B. Kontext- und Initialisierungsparameter, Zugriffsdaten von Datenbanken, Sicherheitseinstellungen und die Pfade zu den Servlets und zu allen HTML- und JSP-Dateien.

Bei der Implementierung eines Servlets wird die Klasse javax.servlet.http.HttpServlet erweitert. Die Klasse HttpServlet enthält Methoden und spezielle Anfrage- und Antwortobjekte, die die Kommunikation zwischen Servlets und Clients vereinfacht. Die Methoden dieser Klasse können verschiedene HTTP-Anfragen (GET, POST, PUT, DELETE, usw.) behandeln. Bei einer Anfrage von einem Client werden Daten an ein Servlet normalerweise über die HTTP-Anfragentypen GET oder POST übertragen. Der Unterschied zwischen HTTP-GET und HTTP-POST liegt an die Übertragungsart der Daten. Bei HTTP-GET werden die Daten an ein Servlet übergeben, indem sie einfach zu der URL hinzugefügt werden. Diese Art von Übertragung soll nur verwendet werden, wenn die Daten nicht so groß sind. Bei HTTP-POST werden die Daten in ein Paket zusammengeführt und an das Servlet versendet.

Die Methoden doGet und doPost von der Klasse HttpServlet bearbeiten die Anfragen und müssen in die abgeleitete Klasse implementiert werden. Beide Methoden enthalten die vom Servlet-Container erzeugten Objekte mit den Daten der Anfrage (Request) und zur Generierung einer Antwort (Response), die als Parameter übergeben werden.

Die folgende Abbildung zeigt den Ablauf einer Web-Anwendung mit Servlets. Zuerst wird ein von einem Benutzer ausgefülltes Formular im Browser abgesendet. Mit der Auslösung der im Formular bestimmten Aktion durch den Browser werden die Formulardaten an den Web-Server gesendet. Der Aktionsname wird vom Web-Server mit der Hilfe des Deployment Descriptor in den Namen einer Servletklasse übersetzt. Die Methode doGet oder doPost der Servletklasse werden mit den Anfragendaten als Parameter aufgerufen. Das Servlet generiert eine Antwort, die der Web-Server an den Browser sendet. Die Antwort wird schließlich vom Browser empfangen und dargestellt.

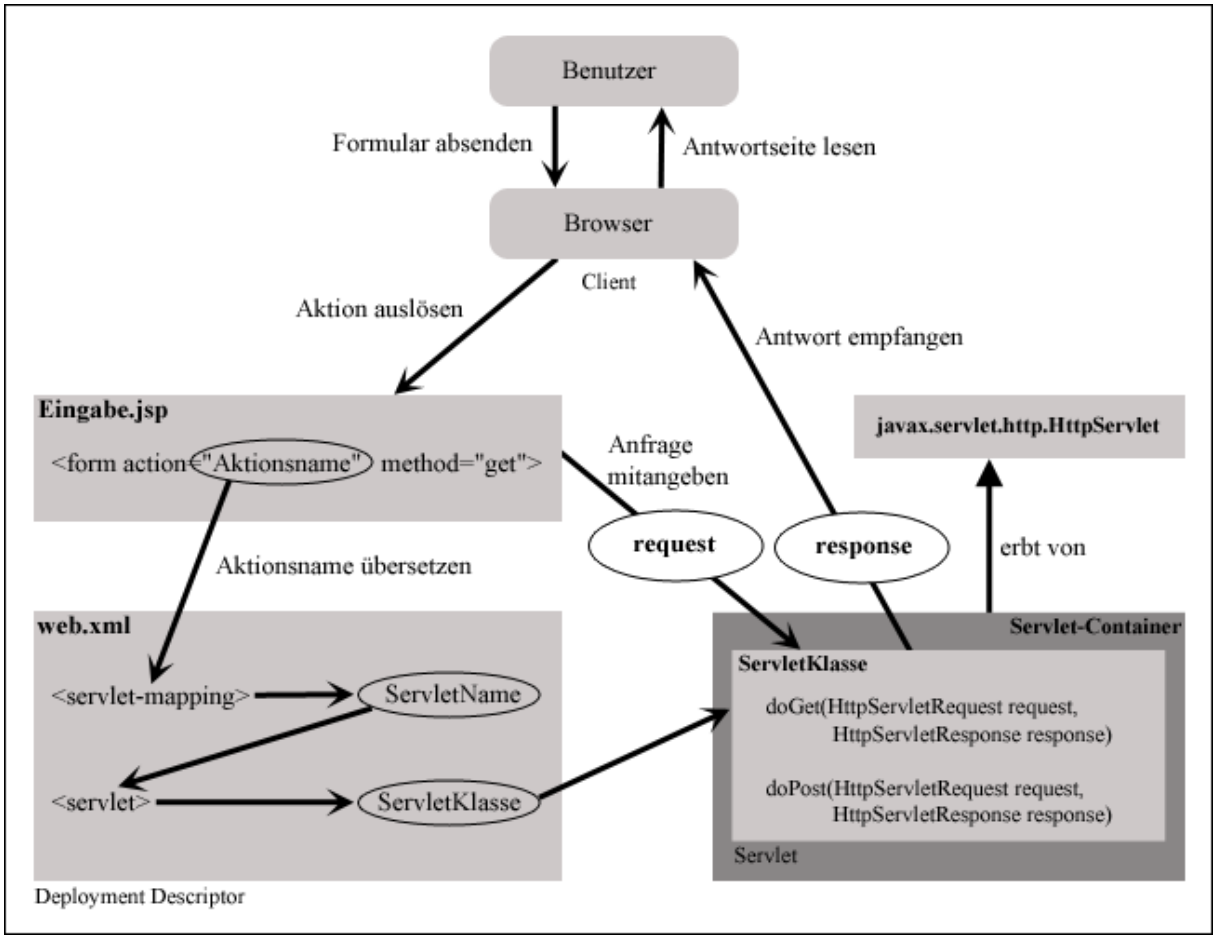

Abbildung 3: Beispiel vom Ablauf einer Web-Anwendung mit Servlets [2]

Die folgende Abbildung zeigt ein Beispiel von der Implementierung des Servlets „HalloServlet“, der die Anfrage von der JSP-Datei aus der Abbildung 2 vom Kapitel „2.1.2 Java Server Pages“ empfängt und beantwortet.

[2] *Vgl. o.V. (2013)*

Die Klasse HalloServlet erweitert die Klasse HttpServlet und enthält die Methoden doGet und doPost. Bei der JSP-Datei werden die Daten in einem Formular dargestellt und über die HTTP-POST-Anfrage abgesendet, somit muss die Methode doPost aufgerufen werden. Die Methode doGet sorgt nur dafür, dass bei einem Programmierfehler in der JSP-Datei immer die richtige Methode doPost mit den Parametern aufgerufen wird.

In der Methode doPost werden zuerst die abgesendeten Daten aus dem Request-Objekt durch die Methode getParameter gelesen. Die Antwort auf die Anfrage wird durch das Response-Objekt abgebildet, mit dem die Methoden setContentType und setCharacterEncoding aufgerufen werden. Die Methode setContentType legt fest, aus welchem Inhaltstyp die generierte Antwort besteht. Mit der Methode setCharacterEncoding kann eine bestimmte Encodierung der Antwort ausgewählt werden, um falsche Darstellung von Zeichen zu vermeiden.

In der Methode doPost gibt dann die Methode getWriter ein Objekt von Typ PrintWriter zurück, in dem die Antwort geschrieben wird. Nachdem die Methode doPost zu Ende ist, wird die Antwort automatisch an den Client gesendet.

```
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class HalloServlet extends HttpServlet {

	protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {

		doPost(request, response);
	}

	protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {

		String name = request.getParameter("name");
		String vorname = request.getParameter("vorname");

		response.setContentType("text/html");
		response.setCharacterEncoding("UTF-8");

		PrintWriter out = response.getWriter();

		out.println("<html>");
		out.println("<body>");
		out.println("<h1>Hallo " + vorname +" "+ name + "</h1>");
		out.println("</body>");
		out.println("</html>");
	}
}
```

Abbildung 4: Beispiel von einem Servlet

2.1.4 Enterprise JavaBeans

Ein Enterprise JavaBean (EJB) ist eine Software-Komponente, die für die Durchführung von Anwendungslogikoperationen auf einem Java-EE-Server verwendet werden. Mit Enterprise JavaBeans soll die Behandlung von wichtigen Merkmalen der Anwendungslogik, wie persistente Datenspeicherung, transaktionale Integrität und Sicherheit vereinfacht werden.

Enterprise JavaBeans werden in drei Arten unterteilt. Ein Entity-Bean dient der persistenten Speicherung von Daten. Ein Session-Bean repräsentiert die Kommunikation zwischen einem Client und einem Server. Die Kommunikation wird durch Anwendungslogikfunktionen dargestellt, für die mehrere Entity-Beans verwendet werden können. Die Message-Driven-Bean ist eine Komponente, die eine asynchrone Kommunikation zwischen EJB-Systemen ermöglicht. Die Nachrichten der Kommunikation werden durch Java Message Service (JMS) bereitgestellt.

Zur Programmierung einer Enterprise JavaBean werden mindestens zwei Schnittstellen und eine Klasse benötigt. Die Schnittstelle Remote legt die Methoden fest, die von einer Client-Anwendung auf der EJB aufgerufen werden können. Die Schnittstelle Home legt die Methoden fest, die dem Erstellen und dem Suchen von EJBs dienen. Dazu kommt auch die Java-Klasse, die die beiden Schnittstellen implementiert. In dieser Klasse werden die EJBs erzeugt.

Die folgende Abbildung zeigt einen Client, das eine EJB mit der Schnittstelle Home erzeugt und dann Bean-Methoden über die Schnittstelle Remote aufruft.

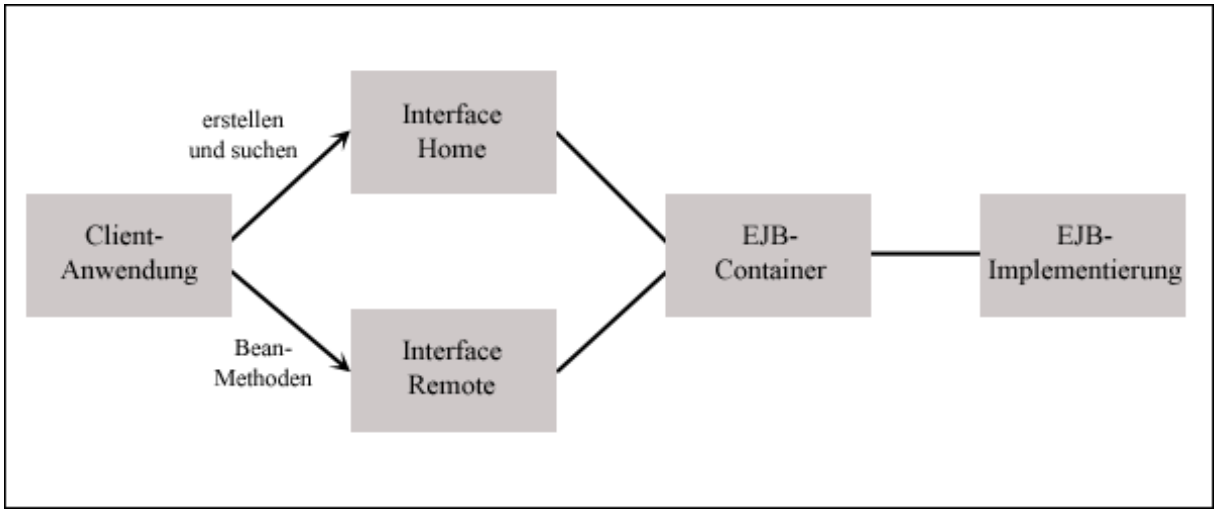

Abbildung 5: Beispiel von einer Anwendung mit EJB [3]

[3] *Vgl. Wutka (2002), S. 125*

2.1.5 Entwicklungsmodelle bei JSP-Anwendungen

Die Softwarearchitektur-Spezifikation von Java-EE schlägt zwei Modelle zur Entwicklung von Web-Anwendungen mit Java Server Pages vor, die Modell 1- und Modell 2-Architektur genannt werden.

Die Modell 1-Architektur ist nur für kleine Web-Anwendungen geeignet. In diesem Modell sind JSP und Servlets für die Kommunikation mit dem Browser zuständig, indem sie Client-Anfragen empfangen und entsprechende Antworten zurücksenden. Die Anwendungslogik der Web-Anwendungen wird von Enterprise JavaBeans verarbeitet.

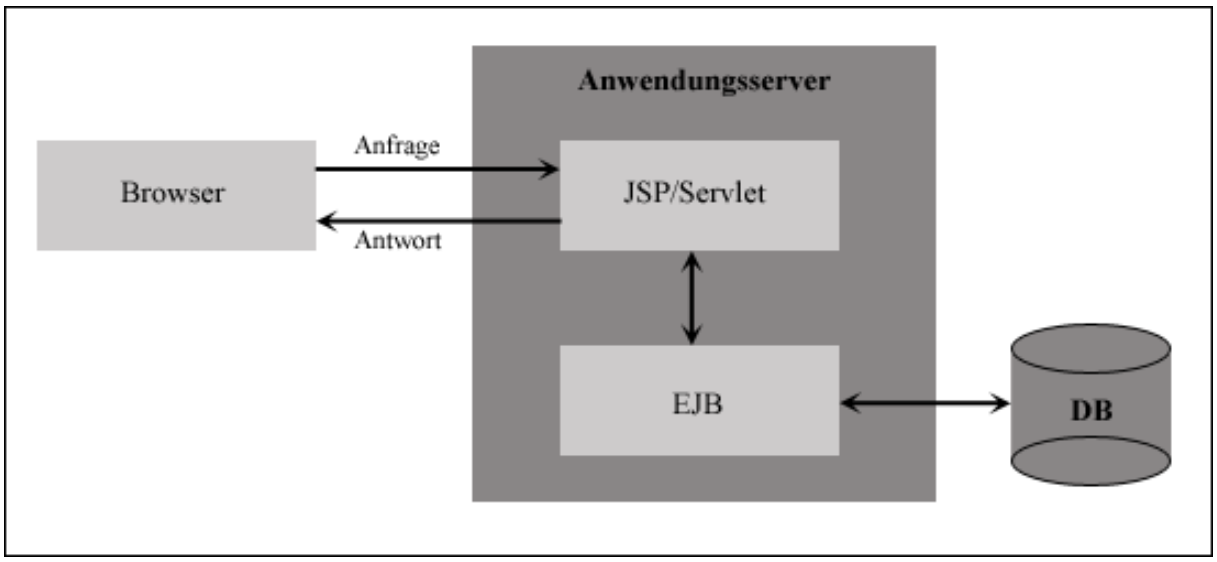

Abbildung 6: Modell 1-Architektur [4]

Die Modell 2-Architektur stellt eine veränderte Version vom Modell-View-Controller-Konzept (MVC-Architektur) dar, das für komplexe Anwendungen entwickelt wurde und eine Trennung zwischen Darstellung (Präsentation) und Inhalt (Daten) vorsieht. Die MVC-Architektur wird durch drei Komponenten realisiert. Das Modell ist für die Anwendungslogik zuständig und enthält die darzustellenden Daten. Bei Änderung der Daten sendet das Modell eine Meldung an das Komponente View. Die View dient der visuellen Darstellung der Daten. Der Controller empfängt die Anfragen der Clients und bestimmt, wie die Anwendung darauf

[4] *Vgl. Rittmeyer (2007)*

reagieren soll. Er führt die Anwendungslogik im Modell aus und legt fest, welche View verwendet werden soll.

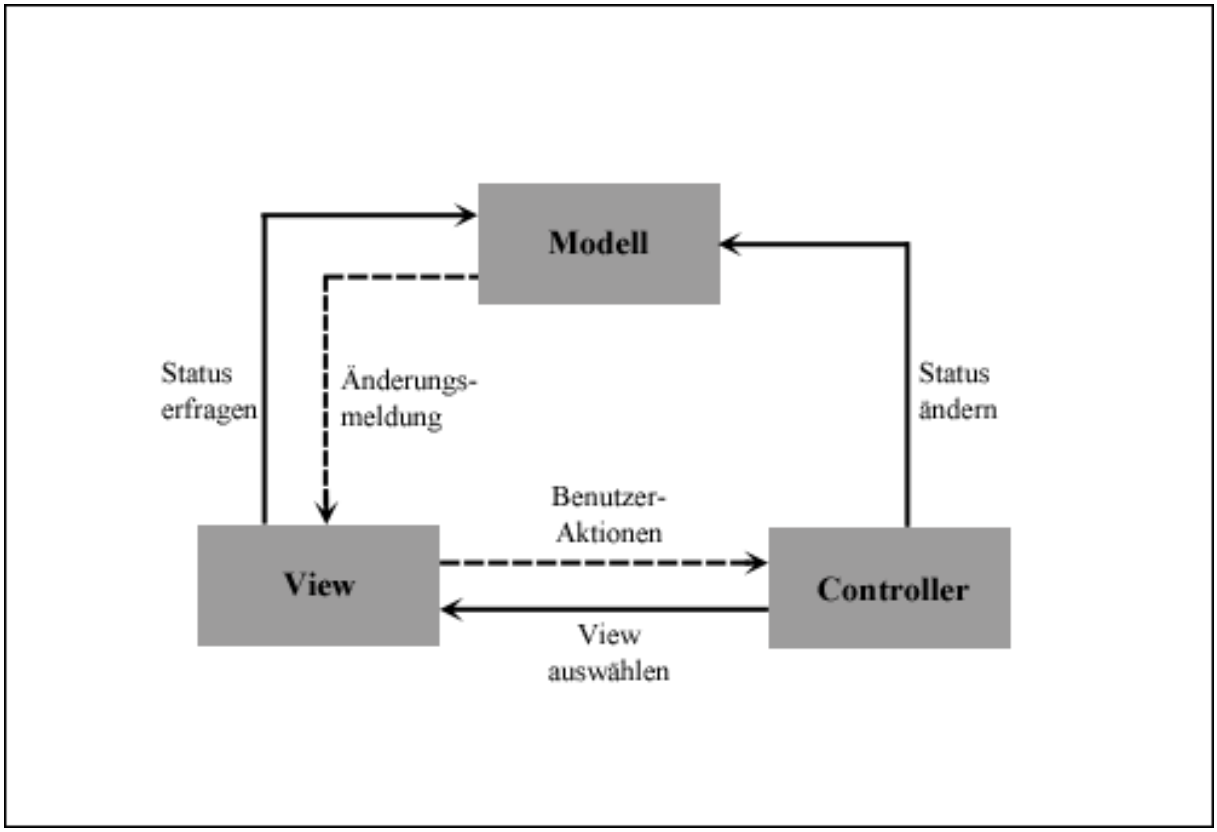

Abbildung 7: MVC-Architektur[5]

Die Modell 2-Architektur wird für die meisten Web-Anwendungen empfohlen. In diesem Modell haben JSPs und Servlets unterschiedliche Aufgaben. Die JSPs sind ausschließlich für die Ausgabe der Antworten verantwortlich, die an die Clients zurückgesendet werden. Die Servlets sind für das Empfangen und Verarbeiten von Anfragen zuständig. Sie erzeugen EJBs, die die Daten darstellen und von JSPs benötigt werden. Die Servlets wählen die JSP aus, die auf die Anfrage beantworten soll.

[5] *Vgl. Turau/Saleck/Lenz (2004), S.16*

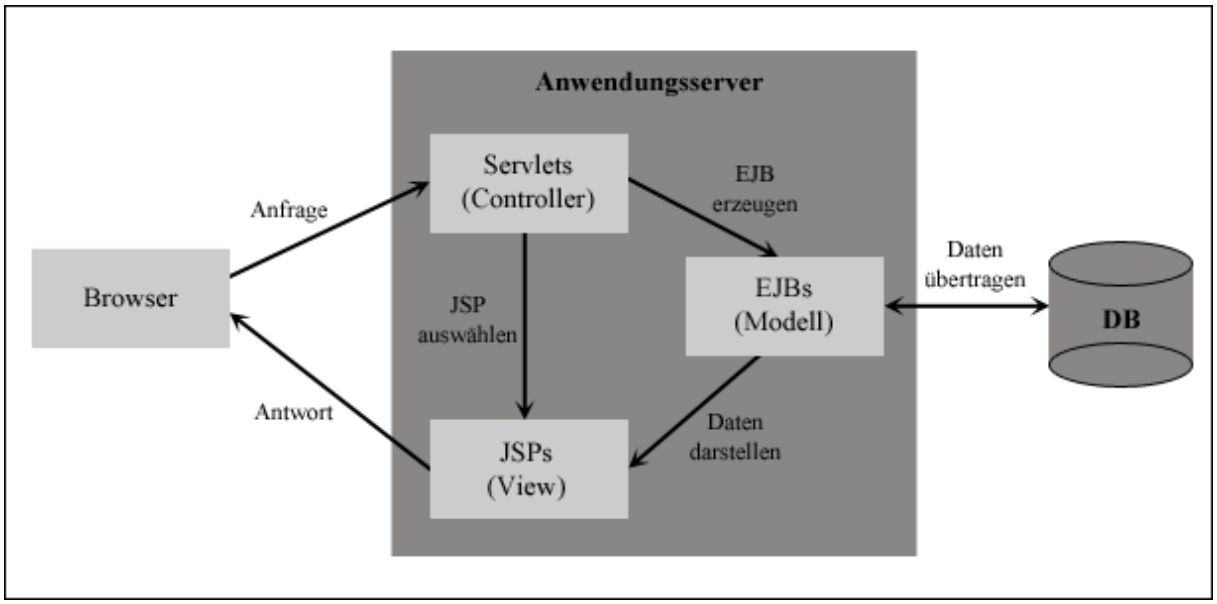

Abbildung 8: Modell 2-Architektur [6]

Das MCV-Konzept kann in einer Web-Anwendung nicht vollständig angewendet werden, weil das Modell die View über Datenänderungen nicht informieren kann. Das Problem liegt in der Web-Kommunikation (HTTP-Protokoll). Die Web-Kommunikation besteht immer aus einer Anfrage vom Client und einer Antwort vom Server. Der Server kann nicht selber aktiv werden und Daten an den Client senden oder diesen über Datenänderungen informieren.

[6] *Vgl. Rittmeyer (2007)*

2.2 DOJO Toolkit

Das Dojo Toolkit ist eine modulare JavaScript-Bibliothek zur Entwicklung von Web-Anwendungen. Mit der Anwendung von Dojo Toolkit soll das Programmieren von modernen Web-Seiten (Rich Internet Application) erleichtert werden, weil dieses Standardfunktionen für Interaktionsmöglichkeiten mit der Oberfläche der Web-Seiten zur Verfügung stellt. Dojo bietet eine Vielzahl an Widgets (Benutzeroberfläche-Komponenten) an, die in Form von vorgefertigten Komponenten bzw. Module aus JavaScript, HTML und CSS in Web-Seiten verwendet werden können. Dojo kann mit verschiedenen Programmiersprachen, wie Perl, PHP oder JSP, eingesetzt werden. Dojo wird von der Dojo Foundation entwickelt und ist als Open-Source verfügbar.

Eine wichtige Eigenschaft von Dojo ist es, dass Dojo wie eine Middleware zwischen der Web-Anwendung und dem vom Benutzer verwendeten Web-Browser fungiert. So kann Dojo bestimmte Merkmale von einzelnen Web-Browsern isoliert, damit die Web-Anwendungen nicht durch Inkonsistenz der Web-Browser beeinträchtigt werden. Mit einer solchen Eigenschaft kann Dojo browserunabhängige Funktionalitäten anbieten, die die Programmierung von Web-Anwendungen vereinfachen. Dadurch wird auch eine hohe Portabilität der Web-Anwendungen ermöglicht.

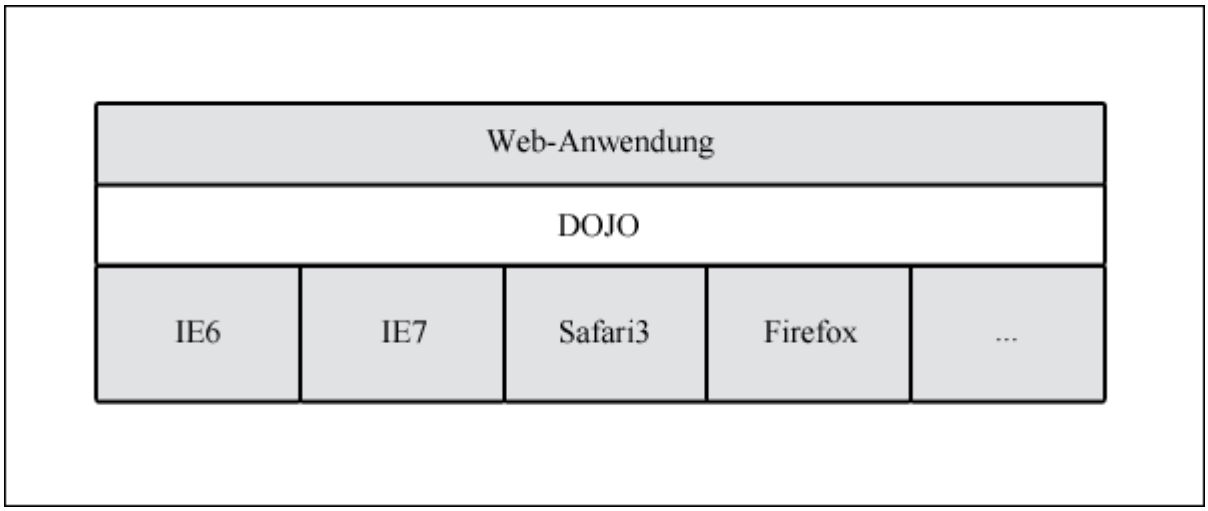

Abbildung 9: Anwendungssichten zwischen Web-Anwendung, Dojo und Web-Browser[7]

[7] *Vgl. Russel (2008), S. 25*

Die Anwendung von Dojo in Web-Seiten bietet folgende Vorteile:

- browserunabhängige Funktionalitäten
- gute Performance
- hohe Portabilität
- Unterstützung der AJAX-Kommunikation
- clientseitige und serverseitige Datenspeicherung
- Funktionen zur Datenverwaltung
- Unterstützung zur dynamischen Gestaltung der Oberfläche
- Unterstützung von Animationen, visuellen Effekten, Grafiken und Diagrammen
- Bereitstellung von vorgefertigten CSS-Themen

2.2.1 DOJO-Architektur

Das Dojo Toolkit stellt ein Paketsystem mit modularem Aufbau und Namensräumen bereit. Die Dojo-Architektur besteht aus drei Hauptpaketen, die Unterpakete und Module enthalten.

Das Paket Dojo stellt den Hauptteil von Dojo Toolkit dar. Dieses Paket wird auch als Core bezeichnet und umfasst viele Module, die die einfache Umsetzung von AJAX-Kommunikation, DOM-Manipulation (Document Object Model), Datenspeicherung, Arrays, Ereignissteuerung, Cookies-Verwaltung, Klasse-Programmierung und Drag-and-Drop-Techniken unterstützen. Dazu beinhaltet dieses Paket auch Internationalisierungs-Bibliotheken im Unterpaket i18n, mit dem man Web-Anwendungen für bestimmte Länder personalisieren kann. So können Texte mit Hilfe von Dictionaries in die jeweilige Sprache des Landes übersetzt und die kulturabhängige Repräsentation von Zahlen, Datum und Währung angepasst werden.

Das Paket Dojo enthält auch die JavaScript-Datei dojo.js, die als Dojo-Loader bezeichnet wird und das Paketsystem von Dojo Toolkit verwaltet. Nachdem diese Datei geladen wurde, ist sie für das Laden von anderen Ressourcen und Modulen des Toolkits zuständig. In früheren Versionen von Dojo hat die Datei dojo.js immer das Paket dojo/_base vollständig geladen, in dem die Hauptmodule mit den wichtigsten Funktionalitäten und Konfigurationsmodule gespeichert waren. Dieses Paket wurde als Base bezeichnet und galt als Basis-Paket von Dojo.

Seit der Version 1.7 von Dojo kann die Datei dojo.js in den asynchronen Modus gesetzt werden, damit sie ausschließlich die in Quellcode angegebenen Ressourcen und Module lädt. So kann eine bessere Performance der Web-Anwendung erzielt werden, indem nur die in der Web-Anwendung erforderlichen Module geladen werden. Aus Kompatibilitätsgründen mit älteren Versionen von Dojo ist immer noch möglich das gesamte Basis-Paket von Dojo automatisch zu laden, dafür muss die Datei dojo.js in Legacy-Modus laufen.

Das Paket Dijit ist eine Dojo-Widget-Bibliothek und enthält vorgefertigte Module, die zur einfachen Erstellung von grafischen Web-Oberflächen dienen. Die Module von Dijit bauen auf die Module des Paketes Dojo. Bei Dijit können HTML-Elemente als Dojo-Typ durch das Attribut „data-dojo-type“ angegeben werden. Zu diesem Attribut können auch Eigenschaften durch das Eigenschafts-Attribut „data-dojo-props“ definiert werden, die das Dojo-Typ-Attribut beschreibt. Diese Elemente können als normales Widget von Dijit betrachtet und ohne Einschränkungen verwendet werden. Die Verwendung von HTML-Elementen mit Dojo-Attribute im Quellcode wird als deklarativ bezeichnet. Dijit verfügt aber auch über viele eigene Widgets, deren Verwendung als programmatisch bezeichnet wird. Einige Widgets von Dijit haben den gleichen Namen von HTML-Elementen, so ist es möglich, ihre Funktionalität einfach zu erkennen. Dijit besitzt auch eine Reihe von eigenen Standard-Themen mit Designs und Farbschemata, die in Web-Seiten eingebunden werden können. Einige Beispiele von Themen sind Tuncra, Soria, Noir und Claro, die auch verändert werden können, indem die entsprechenden CSS-Dateien modifiziert werden.

Das Paket DojoX enthält weitere Komponenten und Module, die die Funktionalität von Module aus Paketen Dojo und Dijit erweitern. Die wichtigen Funktionen, die in diesem Paket ergänzt werden, liegen bei der Datenvisualisierung durch Diagramme, Datenspeicherung, Grafik, Animationen, Kommunikation, Widgets und mobiler Anwendung. Das Paket DojoX wird aber auch für die Entwicklung von zusätzlichen Dojo-Modulen verwendet. Aus diesem Grund sind einige Module in diesem Paket noch in einer experimentellen Phase. Andere Module sind ausgereift und können ohne Einschränkungen verwendet werden. Manche Module wurden sogar aufgegeben und werden nicht weiter entwickelt. Jedes Unterpaket und Modul von DojoX wurde mit einem Zustand versehen, der aus der entsprechenden Readme-Datei entnommen werden kann. Die fünf verschiedenen Zustände lauten „ausgereift“, „experimentell“, „verwaltet“, „veraltet“ und „aufgegeben“. Ab Dojo 2.0 wird es dieses Paket nicht mehr

geben und dessen Module werden entsprechend ihrer Funktionalitäten in die anderen Pakete Dojo oder Dijit integriert.

Zusätzlich zu den drei wichtigen Paketen von Dojo Toolkit gibt es bei der Version SDK (Software Development Kit) auch das Paket Util. Dieses Paket enthält verschiedene Werkzeuge, die die Entwicklung von Web-Anwendung mit Dojo unterstützen. Im Paket Util befinden sich Dienstprogramme und keine Datei, die direkt von einer Web-Seite aufgerufen wird. In der folgenden Tabelle werden die Dienstprogramme vom Paket Util beschrieben.

Dienstprogramm	Beschreibung
Dojo-Build-System	Mit dem Dojo-Build-System können effiziente und optimierte Pakete von JavaScript und CSS erstellt werden. Die Struktur der Pakete kann angepasst und verbessert werden, damit die Performance der Web-Anwendung steigert.
ShrinkSafe	ShrinkSafe ist ein Dienstprogramm zum Komprimieren von JavaScript, das manuell oder auch in Zusammenhang mit dem Dojo Build-System verwendet werden kann.
DOH	DOH ist eine Testumgebung von Dojo, ähnlich wie JUnit. Im Paket util/doh/ befindet sich unter anderem ein DOH-Robot, der für die automatisierten Tastatur- und Maus-Tests verwendet wird.
Checkstyle	Checkstyle ist ein Dienstprogramm zur Überprüfung von Verstößen gegen das Dojo-Styleguide bei JavaScript-Dateien. Dazu gehört auch ein Online-Tool, das die Verstöße automatisch anzeigt.
Dojo-Dokumentations-System	Dojo verwendet eine benutzerdefinierte Inline-Kommentar-Syntax, die als XML strukturiert wird und dazu noch Hinweise auf die offizielle API-Dokumentation erzeugt.
Migration	Das Paket Migration enthält Skripten, die die Migration von älteren Versionen von Dojo zu neueren unterstützen.

Tabelle 1: Dienstprogramme vom Paket Util

Das Dojo Toolkit erlaubt zusätzliche Pakete und Namensräume innerhalb seines Paketsystems, so dass Dojo durch eigene oder von Drittanbietern entwickelte Widgets oder Module erweitert werden kann.

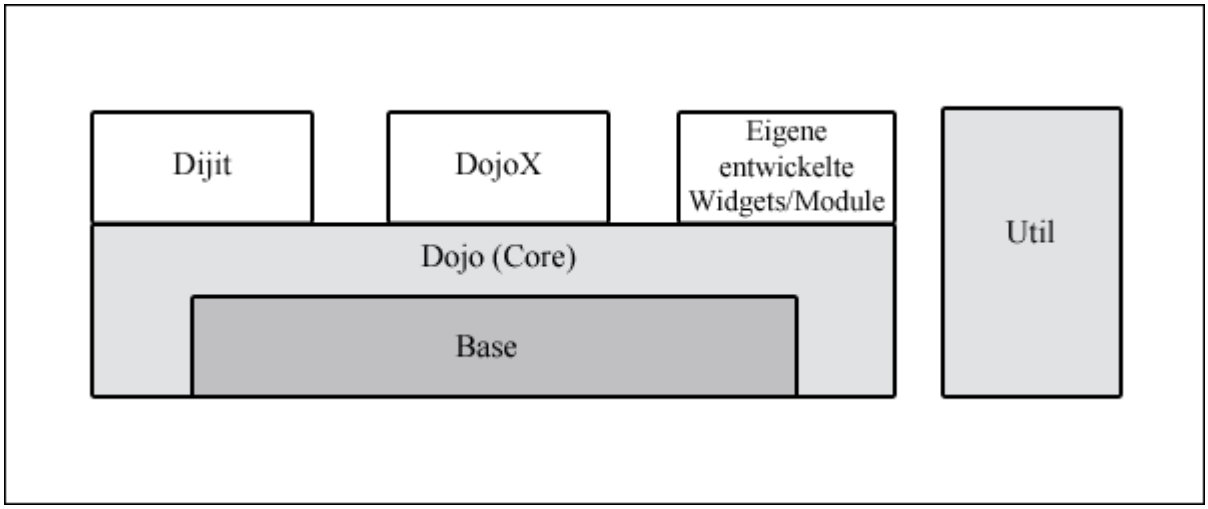

Abbildung 10: Übersicht von der DOJO-Architektur [8]

2.2.2 Paketsystem und Namensräume von DOJO

Das Paketsystem von Dojo ist in einzelne Pakete und Unterpakete gegliedert, in denen Module und Widgets gespeichert sind. Daraus entstehen die Namensräume von Dojo, die eindeutige Pfadnamen zu Modulen und Widgets darstellen. Zum Beispiel bildet sich der Pfadname des Moduls dojo/_base/config aus dem Hauptpaket dojo, das das Paket _base enthält. Dort befindet sich die JavaScript-Datei config.js, die das Modul ist.

Mit Namensräumen können Funktionalitäten und Ressourcen besser getrennt werden und Namenskonflikte vermieden werden. Der Gültigkeitsbereich beschränkt sich auch nur im umgebenden Namensraum. Die Namensräumen-Technik wird in vielen modernen objektorientierten Programmiersprachen und auch in XML eingesetzt.

[8] *Vgl. Russel (2008), S. 3*

Die Namensräume von Dojo beginnen entweder mit dojo, dijit oder dojox, die den drei Hauptpaketen entsprechen. In den folgenden Tabellen wird eine Übersicht über die wichtigsten Namensräumen in Dojo Toolkit mit den bereitgestellten Funktionalitäten gezeigt.

Dojo-Pakete/-Module	Beschreibung
dojo/_base	Dieses Paket und die zugehörigen Module entsprechen der Funktionalität des Basis-Pakets (Base) von älteren Versionen von Dojo.
dojo/_base/Color	Color ist eine Basis-Klasse, die eine einfache Speicherung von Farben im RGBA-Farbmodell ermöglicht.
dojo/_base/config	Mit diesem Paket kann Dojo durch das dojoConfig-Objekt konfiguriert werden.
dojo/_base/connect	Dieses Modul enthält Funktionen, die Methoden, Ereignisse und Themen verbinden.
dojo/_base/declare	Das Modul dojo/_base/declare enthält Funktionen zu Dojo-Klassen, mit denen die objektorientierte Konzepte unterstützt werden.
dojo/_base/fx	Dieses Modul enthält grundlegende Animations-Funktionen.
dojo/_base/html	Zur Abwärtskompatibilität definiert das Modul dojo/_base/html Aliasnamen für DOM- und HTML-Funktionen.
dojo/_base/kernel	Dieses Paket funktioniert als Ladeprogramm für die Erzeugung der Namensräume von Dojo.
dojo/colors	Colors ist ein Modul, das die Basisklasse dojo/_base/Color mit zusätzlichen Methoden und benannten Farben erweitert.
dojo/cookie	Dieses Modul ermöglicht den Umgang mit clientseitigen Cookies.
dojo/data	Das Paket dojo/data stellt eine einheitliche Datenzugriffsschicht dar, die die Konzepte von Datenbank-Treiber, Service-Endpoints und einzigartigen Datenformaten ersetzt. Das Paket dojo/store wird die Aufgaben von dojo/data in die nächste Version vom Dojo übernehmen.

dojo/date	Dieses Modul bietet verschiedene Methoden zur Manipulation eines Datums.
dojo/dom	Dieses Modul definiert die Hauptfunktionen von Dojo-DOM-API. Die Standard-Rückgabevariable für dieses Modul ist ein DOM-Objekt, das die HTML-Dokument darstellt..
dojo/dom-form	Dieses Modul stellt Bearbeitungsfunktionen für Formulare zur Verfügung.
dojo/domReady!	Das Modul dojo/domReady! verhindert, dass eine Web-Seite geladen wird, bevor das DOM-Objekt mit allen Modulen vollständig geladen wurde.
dojo/html	Das Modul dojo/html stellt wichtige Werkzeuge zur Verwaltung von HTML-Elementen.
dojo/i18n	Dieses Modul ermöglicht die Internationalisierung-Features von Dojo. Es arbeitet als Plugin für das Laden von Text-Ressourcen.
dojo/json	Dieses Modul dient der Parsen und Serialisierung von JSON-Objekten.
dojo/on	Das Modul dojo/on ist ein Allzweck-Event-Handler-Modul für DOM-Knoten und andere Event-Objekte. Dieses Modul stellt verschiedene Funktionen zur Bearbeitung von DOM-Knoten bereit.
dojo/parser	Das Modul dojo/parser konvertiert spezielle Knoten im DOM-Objekt und wandelt sie in Widgets oder andere Objekte um.
dojo/query	Das Paket dient der Verwaltung von DOM-Objekten, indem DOM-Knoten ausgewählt und bearbeitet werden können.
dojo/request	Dieses Paket bietet Module für eine Anfrage bei einer asynchronen Kommunikation zwischen Client und Server (AJAX-Kommunikation).
dojo/request/xhr	Das Modul dojo/request/xhr verwendet das XMLHttpRequest-Objekt (XHR), um eine asynchrone Kommunikation zwischen Client und Server zu ermöglichen.

dojo/store	Mit diesem Paket wird eine einheitliche Schnittstelle für den Zugriff und Manipulation der gespeicherten Daten zur Verfügung gestellt.
dojo/store/DataStore	Das Modul DataSore funktioniert wie ein Adapter für die Verwendung des alten Paketes dojo/data mit dem neuen Paket dojo/store.
dojo/string	Mit dem Modul dojo/string stehen einige Funktionen zur einfachen String-Manipulation zur Verfügung.
dojo/text	Mit diesem Modul können beliebige Zeichenketten aus einer Text-Datei geladen und zurückgegeben werden.

Tabelle 2: Namensräume im Dojo-Paket

Dijit-Pakete/-Module	Beschreibung
dijit/_Container	_Container enthält eine Liste von untergeordneten Widgets (Container-Elemente, wie dijit/layout/BorderContainer und dijit/layout/TabContainer) und kann für diese als Superklasse verwendet werden.
dijit/_MenuBase	_MenuBase ist eine Basis-Klasse, die von dijit/Menu und dijit/MenuBar verwendet werden kann.
dijit/_WidgetBase	WidgetBase ist die Basisklasse für alle Widgets im Dijit-Paket.
dijit/Menu	Mit diesem Widget können Kontext-Menüs für Web-Oberflächen erstellt werden.
dijit/MenuBar	Dieses Widget dient der Erstellung von horizontalen Menüleisten für die Web-Oberfläche.
dijit/DropDownMenu	Das DropDownMenu-Widget ist ein vertikales Menü, das in Verbindung mit den Widgets dijit/form/ComboButton, dijit/formDropDownButton und dijit/MenuBars verwendet werden kann.

dijit/Calendar	Dieses Widget stellt einen Kalender mit verschiedenen Funktionen dar.
dijit/ColorPalette	Mit diesem Widget wird eine Farbpalette auf einer Web-Oberfläche erzeugt.
dijit/Editor	Das Widget Editor bietet eine komplette Editor-Oberfläche zum Schreiben und zur Bearbeitung von Texten an, wie das Programm Word.
dijit/ProgressBar	Mit ProgressBar wird eine dynamische Fortschrittsanzeige einer laufenden Operation dargestellt. Der Fortschritt werden durch Aufrufe von JavaScript-Funktionen aktualisiert.
dijit/Tree	Mit diesem Widget können Bäume erstellt werden, die zur Gliederung von hierarchischen Listen dienen.
dijit/form	Dieses Paket enthält wichtige Elemente, um ein Formular auf Web-Oberflächen aufzubauen.
dijit/form/Button	Dieses Widget bildet einen normalen Button ab.
Dijit/form/CheckBox	Dieses Widget entspricht dem CheckBox-Element von HTML mit dem Unterschied, dass das Widget verschiedene Styles annehmen kann.
dijit/form/ComboButton	Dieses Widget stellt einen Button mit einer Drop-Down-Funktion dar.
dijit/form/DateTextBox	DateTextBox ist ein Eingabefeld, das eine einfache Datumseintragung ermöglicht.
dijit/form/Select	Dieses Widget entspricht dem Select-Element von HTML.
dijit/form/Textarea	Dieses Widget ähnelt dem Texarea-Element von HTML. Der Unterschied liegt darin, dass sich das Widget mit Einfügung von Texte dynamisch vergrößert
dijit/icons	Das Paket dijit/icons enthält CSS- und Bilder-Dateien, die zur Darstellung von grafischen Symbolen in Web-Anwendungen dient.
dijit/layout	Das Paket dijit/layout besitzt eine Vielzahl an Layouts für Web-Oberflächen, die mit anderen Layouts kombiniert werden können.

dijit/layout/_LayoutWidget	_LayoutWidget ist eine Basisklasse, von der andere Layout-Widget erben. In dieser Klasse werden die Methoden addChild(), removeChild(), start() und resize() implementiert.
dijit/layout/LayoutContainer	Dieses Widget ist ein Container, der in bis zu fünf Regionen aufgeteilt werden kann. Diese Regionen enthalten entweder einen anderen Container oder ein ContentPane, auf dem Elemente und andere Widgets platziert werden.
dijit/layout/BorderContainer	Der BorderContainer ist ein LayoutContainer mit der zusätzlichen Möglichkeit, die Größe der ContentPanes einfach mit dem Ziehen des Mauszeigers zu verändern.
dijit/layout/TabContainer	Der TabContainer kann viele ContentPanes enthalten, die als Registerkarte dargestellt werden.
dijit/layout/ContentPane	ContentPane ist die letzte Komponente in der Layout-Hierarchie und enthält die Elemente, die auf der Web-Oberfläche angezeigt werden.
dijit/themes	Dieses Paket enthält verschiedene vorgefertigte CSS-Dateien, die zum visuellen Styling von Widgets und Web-Seiten verwendet werden können.

Tabelle 3: Namensräume im Dijit-Paket

DojoX-Pakete/-Module	Beschreibung
dojox/analytics	Dieses Paket stellt ein analytisches Client-Überwachungs-system dar, das aus dem Basis-Analytik-System und einer beliebigen Anzahl von Plugins besteht. Mit diesen Plugins kann die Übertragung von Daten zwischen Client und Server protokolliert werden.
dojox/app	Das Modul dojox/app ist ein Anwendungs-Framework für mobile Geräte, um die Konfiguration von Anwendungen mit verschachtelten Views zu vereinfachen und den Übergang zwischen den Views zu erleichtern. Die Konfiguration der Anwendungen erfolgt über eine Konfigurationsdatei.

dojox/charting	Das Paket dojox/charting ist eine Bibliothek, die eine große Anzahl von Funktionen zur dynamischen Erstellung von Diagrammen und Grafiken enthält. Für Diagramme müssen Plots, Achsen und Daten definiert werden. Die Plots beschreiben, wie die Daten visualisiert werden sollen. Die Achsen stellen die Dimensionen der Daten dar, die auf dem Diagramm gezeigt werden.
dojox/charting/plot2d	Das Paket dojox/charting/plot2d enthält Module für die Erstellung von zwei dimensionalen Diagramme (2D-Diagramme).
dojox/charting/plot2d/Bar	Mit dem Modul dojox/charting/plot2d/Bar können zwei dimensionale Stabdiagramme dargestellt werden.
dojox/charting/plot2d/Columns	Dieses Modul dient der Erzeugung von zweidimensionalen Säulendiagrammen.
dojox/charting/plot2d/Lines	Mit diesem Modul können zweidimensionale Kurvendiagramme erstellt werden.
dojox/charting/plot2d/Pie	Mit dem Modul dojox/charting/plot2d/Pie Kreisdiagramme erstellt werden
dojox/charting/plot3d	Das Paket dojox/charting/plot3d enthält Module für die Erstellung von dreidimensionalen Diagrammen (3D-Diagramme).
dojox/charting/themen	Dieses Paket enthält verschiedene vorgefertigte Module, die zum visuellen Styling von Diagrammen verwendet werden.
dojox/collections	Dieses Modul stellt Implementierungen für verschiedene Collection-Klassen wie ArrayList, Dictionaries, Queue, SortedList oder Stack zur Verfügung.
dojox/color	Dieses Paket enthält Module mit Farbmethoden und einem Farbgenerator, die das Paket dojo/colors erweitern.
dojox/fx	Mit diesem Paket können Animationen durch animierte Übergänge von CSS-Klassen-Knoten erstellt werden, indem der Klassennamen eines Knotens hinzugefügt, entfernt oder umgeschaltet wird.

dojox.gfx	Das Paket dojox.gfx stellt eine Schnittstelle für das Zeichnen von browserübergreifenden Grafiken dar.
dojox/html/styles	Mit diesem Modul können CSS-Elemetne dynamisch erstellt, entfernt oder geändert werden.
dojox/image	Das Paket dojox/image enthält eine Reihe von Bild-bezogenen Widgets als Erweiterungen vom Dijit-Paket.
dojox/json	Dieses Paket enthält viele Werkzeuge zur Bearbeitung von JSON und anderen Objektdaten-Strukturen.
dojox/layout	Dieses Paket beinhaltet viele Widgets zur Gestaltung von Layouts auf Web-Seiten.
dojox/validate	Dieses Paket stellt eine Reihe von Funktionen zur Verfügung, die die Validierung von Eingaben automatisch überprüfen. Solche Funktionen sind für Formulare sehr nützlich, z. B. bei der Eingabe von einer E-Mail-Adresse oder Telefonnummer.
dojox/widget	Dieses Paket ist eine Sammlung von verschiedenen, nicht kategorisierten Widgets.
dojox/xml	Dieses Paket enthält unterschiedliche Module zur Verwaltung von XML-Strukturen. Diese Module sollen Zugriff auf verschiedene Pakete haben, um Code-Duplizierung in Web-Anwendungen zu beseitigen. Derzeit beinhaltet das Paket einen nativen JS-DOMParser, der XML zu einem JS-Objekt-Baum umwandelt.

Tabelle 4: Namensräume im DojoX-Paket

2.2.3 Laden von DOJO

Bei dem Aufruf einer Web-Seite muss zuerst die JavaScript-Datei dojo.js (Dojo-Loader) geladen werden, damit andere Ressourcen und Module vom Dojo geladen und verwendet werden können. Es gibt zwei Methoden, um den Dojo-Loader in einer Web-Anwendung zu laden.

Die einfachste Methode ist ein Content Delivery Network (CDN) zu nutzen. Ein CDN ist ein Netzwerk von verteilen Servern, das wie ein Distributionssystem funktioniert. Die Server stellen Inhalte zur Verfügung, auf die von anderen Anwendungen zugegriffen werden können. Verschiedene Unternehmen, wie AOL und Google, ermöglichen das Laden vom Dojo-Loader und von Modulen über ein CDN. Diese Methode soll aber nur in kleinen Web-Anwendungen verwendet werden, die nicht so viele Module benötigen. Der Grund dafür ist, dass Probleme beim Laden von vielen Modulen aus einer externen Quelle auftreten können. Das führt zu einer Verlangsamung des Aufrufs und der Darstellung von der gesamten Web-Seite.

Der URL zum Dojo-Loader muss in einem Script-Element im HTML-Dokument stehen. Mit dem Attribut data-dojo-config kann das Laden von Modulen konfiguriert werden. Der Wert „async: true“ gibt an, dass asynchrone Übertragungen durchgeführt und nur die im Quellcode angeforderten Module geladen werden. Diese Konfiguration gilt als Standard-Einstellung und soll seit Dojo 1.7 immer verwendet werden. Das Script-Element wird häufig nicht im Head-Teil, sondern im Body-Teil des HTML-Dokuments eingefügt. Damit soll verhindert werden, dass die Web-Seite (HTML-Elemente) fertig geladen und gerendert wird, bevor die Module von Dojo geladen wurden.

In der folgenden Abbildung wird dargestellt, wie Dojo über ein CDN geladen wird.

```
<!DOCTYPE html>
<html lang="de">
<head>
   <meta charset="utf-8">
   <title>Dojo laden</title>
</head>
<body>
   <!-- dojo wird geladen -->
   <script src="http://ajax.googleapis.com/ajax/libs/dojo/1.9.1/dojo/dojo.js"
           data-dojo-config="async: true"></script>
</body>
</html>
```

Abbildung 11: Laden des Dojo-Loaders von einem CDN

Bei der zweiten Methode erfolgt das Laden des Dojo-Loader direkt vom Web-Server. Dafür muss das Dojo Toolkit auf dem Server installiert sein und der Zugriff durch die Web-Anwendung muss erlaubt sein.

Die folgende Abbildung zeigt den Quellcode zum Laden des Dojo-Loaders vom Web-Server.

```
<!DOCTYPE html>
<html lang="de">
<head>
   <meta charset="utf-8">
   <title>Dojo laden</title>
</head>
<body>
   <!-- dojo wird geladen -->
   <script src="dojo-1.9.1/dojo/dojo.js"
           data-dojo-config="async: true"></script>
</body>
</html>
```

Abbildung 12: Laden des Dojo-Loaders vom Web-Server

2.2.4 Konfiguration von DOJO

Das Dojo Toolkit stellt einfache und komfortable Möglichkeiten zur Verfügung, um Konfigurationen vorzunehmen. Bei der Entwicklung von Web-Anwendungen mit Dojo können bestimmte globale Einstellungen angepasst werden, um die Arbeit von Dojo zu steuern. Somit können z. B. Dojo-Komponenten umbenannt werden, oder Dojo-Ressourcen in anderen Nicht-Standard-Paketen platziert werden.

Bei Dojo werden Einstellungen durch ein Konfigurationsobjekt (dojo/_base/config) vorgenommen, das erst während des Dojo-Laden-Prozesses definiert wird. Das Konfigurationsobjekt wird durch das dojoConfig-Objekt initialisiert, indem Werte den Eigenschaften (Properties) des dojoConfig-Objekts zugewiesen werden. Während der Ausführung von der Web-Anwendung können die Eigenschaften des dojoConfig-Objekts auch verändert werden und neue Werte erhalten. Bei Dojo ist es auch möglich, eigene Eigenschaften zum dojoConfig-

Objekt zusätzlich zu deklarieren, die in der entwickelten Web-Anwendung verwendet werden. Es gibt drei Methoden, um Dojo zu konfigurieren.

Die erste Methode erfolgt durch die Angabe der Eigenschaften in das Attribut data-dojo-config im HTML-Dokument. Dieses Attribut muss sich im gleichen Script-Element befinden, in dem das Laden von Dojo durchgeführt wird. Die Eigenschaften des Attributs erhalten bestimmte Werte und werden durch ein Komma getrennt. Das Attribut data-dojo-config wurde in Dojo 1.6 eingeführt und ist ein spezifisches HTML5-Attribut, das nur in HTML5-Dokumenten erkannt werden kann. Diese Methode funktioniert aber nicht richtig, wenn die Web-Seiten über einen Proxy-Server mit HTML-Rewrite-Funktion umgeleitet werden.

Im folgenden Beispiel wird gezeigt, wie das Attribut data-dojo-config aufgebaut ist und wie Werte den Eigenschaften zugewiesen werden. Die Eigenschaft „async“ mit dem Wert „true“ schaltet den asynchronen Modus ein. Mit „parseOnLoad: true“ werden Dijit-Widgets und der eingebettete Code beim Seitenladen automatisch geparst. Debug-Informationen können mit der Eigenschaft „isDebug“ ein- und ausgestellt werden, die bei der Entwicklung von Web-Anwendungen wichtig sind. Mit der Eigenschaft „locale“ können spezifischer i18n- und Orts-einstellungen vorgenommen werden, damit Ressourcen für ein bestimmtes Land und eine bestimmte Sprache geladen werden.

```
<!DOCTYPE html>
<html lang="de">
<head>
   <meta charset="utf-8">
   <title>Dojo </title>
</head>
<body>
   <script src="dojo-1.9.1/dojo/dojo.js"
           data-dojo-config="async: true,
                             parseOnLoad: true,
                             isDebug: true,
                             locale: 'de-de'"></script>
</body>
</html>
```

Abbildung 13: Konfiguration von Dojo durch das Attribut data-dojo-config

Bei der zweiten Methode zur Dojo-Konfiguration wird das dojoConfig-Objekt explizit erzeugt und mit den Eigenschaften initialisiert, die in geschweifte Klammern eingeschlossen werden. Diese Methode soll verwendet werden, wenn zu viele Eigenschaften benötigt werden.

Das dojoConfig-Objekt wird in einem Script-Element deklariert, das vor dem Script-Element zum Laden von Dojo (dojo.js) stehen muss. Der Grund dafür ist, dass das dojoConfig-Objekt initialisiert werden muss, bevor Dojo geladen wird.

Im folgenden Beispiel wird die gleiche Konfiguration von Dojo aus dem vorherigen Beispiel vorgenommen.

```
<!DOCTYPE html>
<html lang="de">
<head>
   <meta charset="utf-8">
   <title>Dojo </title>
</head>
<body>
   <script>
         var dojoConfig = {
                  async: true,
                  parseOnLoad: true,
                  isDebug: true,
                  locale: 'de-de'
         };
   </script>
   <script src="dojo-1.9.1/dojo/dojo.js "></script>
</body>
</html>
```

Abbildung 14: Konfiguration von Dojo durch das dojoConfig-Objekt

Bei der dritten Methode kann das dojoConfig-Objekt in einen benutzerdefinierten Build-Prozess über den Parameter scopeDjConfig aufgenommen werden. Bei Dojo können benutzerdefinierte Build-Prozesse durch die Skripte build.sh (UNIX) oder build.bat (Windows) aus dem Paket util/buildscripts ausgelöst werden. Diese werden normalerweise in Konsolen ausgeführt, wie das folgende Beispiel zeigt. Diese Methode kann aber nur bei einem Build-Prozess verwendet werden.

```
./build.sh scopeDjConfig = { isDebug:true }
```

Abbildung 15: Konfiguration von Dojo durch den ScopeDjConfig-Parameter

In der folgenden Tabelle werden die Eigenschaften vom dojoConfig-Objekt aufgelistet, mit denen globale Einstellungen von Dojo vorgenommen werden können.

Eigenschaften	Werte	Beschreibung
addOnLoad	addOnLoad = [“do-jo/parser”, function(){…}];	addOnLoad sichert, dass die Funktion erst aufgerufen wird, nachdem die Module vollständig geladen wurden. addOnLoad wird seit Dojo 1.7 nicht mehr verwendet.
afterOnLoad	true oder false (default)	Wenn diese Eigenschaft auf true gesetzt ist, können Module oder Ressourcen direkt nach dem Laden vom Dojo-Loader (dojo.js) geladen. So kann man beeinflussen, welche Module oder Ressourcen zuerst geladen werden. Dafür muss der verwendete Dojo-Loader selbst erstellt werden und darf nicht von dojo/dojo.js stammen.
async	true oder false (default)	Mit dieser Eigenschaft kann festgelegt werden, ob die Daten-Übertragung asynchron durchgeführt werden soll. Seit Dojo 1.7 soll diese Eigenschaft immer auf true gesetzt werden.
baseUrl:	“/js/dojo/”	Mit dieser Eigenschaft kann der Hauptteil eines URL zu einem Modul oder Dojo-Paketen definiert werden.

cacheBust	true oder false (default)	Mit dieser Eigenschaft kann das Modul-Caching vermieden werden.
callback	callback: function(Modul) {…}	Mit callback können die bei deps geladenen Module verwendet werden.
debugContainerId	ID der Container (Div-Elemente:)	Mit debugContainerId können die IDs von Div-Elementen auf das Konsolenfenster des Web-Browsers ausgegeben werden, damit diese im Quellcode verwiesen werden, um Fehler schneller zu finden.
defaultDuration	200	Mit dieser Eigenschaft kann man entscheiden, wie lang (in Milli-sekunden) eine Dijit-Animation standardmäßig dauern soll.
deps	deps: ["dojo/parser"]	Mit deps kann man festlegen, welche Module unmittelbar nach dem Dojo-Loader geladen werden
dojo-firebug	true oder false (default)	Mit dojo-firebug kann das Web-Entwicklungstool Firebug eingebunden und verwendet werden. Das Tool Firebug muss in dem Web-Browser schon integriert sein.
dojo-debug-messages	true oder false (default)	Mit dojo-debug-messages können Debugg-Informationen von einer Web-Anwendung erzeugt und ausgegeben werden. Die Eigenschaft isDebug muss auf true gesetzt werden.
dojoBlankHtmlUrl	dojo/resources/blank.html	Diese Eigenschaft definiert eine leere IFrame, die von einigen Modulen (z. B. dojo/io/iframe) benötigt werden, um sicherzustellen, dass native Steuerelemente nicht durch Popups beeinträchtigt werden.

extraLocale	[“ja-jp”], [“de-de”], …	Mit der Eigenschaft locale können zusätzliche Dateien zu lokalen Benutzer-Einstellungen geladen werden, mit denen eine dynamische Umschaltung der Sprache ermöglicht wird. Solche Funktionalität wird bei mehrsprachigen Web-Seiten verwendet.

has	has: {
	"dojo-firebug": true
	}
		Mit has können bestimmte Features von Dojo konfiguriert werden.

isDebug	true oder false (default)	Bei dieser Eigenschaft werden Debugg-Tools geladen, die die Debugg-Informationen auf dem Debugg-Konsolenfenster des Web-Browsers ausgeben.
locale	“en-us”, “de-de”, …	Mit dieser Eigenschaft können die Sprache und das Land vom Benutzer der Web-Anwendung festgelegt werden. So werden entsprechende Dateien geladen und bestimmte lokale Benutzer-Einstellungen vorgenommen. Falls in dieser Eigenschaft kein Wert zugewiesen wird, dann legt Dojo den Wert anhand des Browser-Navigator-Objekts selbst fest. Nachdem Dojo geladen wurde, kann der Wert dieser Eigenschaft nicht mehr verändert werden.

maps	map: {
	dijit1.7: {
	dojo: "dojo1.7"
	}
	}
		Mit map können Pfadnamen zu verschiedenen Pfadnamen zugeordnet werden.

packages	packages: [{
	name: "modul1",
	location: "/js/modul1"
	}, {
		Die Eigenschaft packages besteht aus einem Array von Objekten mit den Namen und Pfadnamen von Modulen,

	name: "modul2", location: "/js/modul2" }]	die von Dojo-Loader geladen werden sollen.
parseOnLoad	true oder false (default)	Mit parseOnLoad kann man definieren, dass Dijit-Widgets und der eingebettete Code beim Seitenladen automatisch geparst werden. Es wird aber empfohlen, diese Eigenschaft nicht auf true zu setzen, sondern sie einfach vom dojoConfig-Objekt wegzulassen. Stattdessen soll das Modul dojo/parser durch require(“dojo/parser”) geladen werden, mit dem die Funktion parser.parse() aufgerufen wird.
paths	{“Modul”: ”../js/modul”}	Diese Eigenschaft legt den Pfadnamen von benutzerdefinierten Module oder Widgets fest, die geladen und verwendet werden können.
require	require(“dojo/parser”);	Mit require kann festgelegt werden, welche Module unmittelbar nach dem Dojo-Loader geladen werden. require wird in dieser Form seit Dojo 1.7 nicht mehr verwendet.
skipIeDomLoaded	true oder false (default)	Diese Eigenschaft ist nur für den Web-Browser Internet Explorer spezifiziert. Damit kann man die Ausführung des DOMContentLoaded-Ereignisses überspringen. Der Grund für diese Überspringung ist, dass ältere Versionen von Internet Explorer dieses Ereignis nicht unterstützen. Das DOMContentLoaded-Ereignis soll erst ausgeführt werden, nachdem der Parser den gesamten HTML-Code eingelesen hat und den

		kompletten DOM-Baum zugänglich gemacht hat.
transparentColor	[255,255,255]	Der Wert dieser Eigenschaft besteht aus einem Array mit drei Zahlen, die den drei RGB-Farbkomponenten des RGB-Farbmodells entsprechen. Diese Zahlen geben an, wie die transparente Farbe in dojo/Color dargestellt werden soll. Der Default-Wert ist [255, 255, 255], der der Farbe Weiß entspricht.
waitSeconds	z. B. waitSeconds: 5	Mit dieser Eigenschaft kann definiert werden, wie lang auf das Laden eines Moduls gewartet werden soll. Default-Wert ist 0, d. h. es soll gewartet werden, bis das Modul vollständig geladen wurde.

Tabelle 5: Eigenschaften vom dojoConfig-Objekt

2.2.5 Module erstellen und laden

Seit Dojo 1.7 wurde die Asynchronous Module Definition (AMD) als Standard für die JavaScript-Module von Dojo eingeführt. Die AMD ist ein Format, das die Erzeugung und das asynchrone Laden von Modulen und die Bestimmung von Abhängigkeiten zwischen den Modulen ermöglicht.

Ein Paket von Dojo kann aus mehreren Modulen bestehen, die von anderen Paketen oder Modulen abhängen können. Die Module und Pakete, die in dieser Abhängigkeit stehen, müssen mitgeladen werden. Im Gegensatz zu früheren Versionen von Dojo, die das Basis-Paket automatisch geladen haben, kann die AMD durch das Paketsystem von Dojo und seine Namensräume nur die Module und Pakete laden, die im Quellcode benötigt werden. Beim Erstellungsprozess einer Dojo-Anwendung wird eine Liste von Paketen und Module eingelesen und

daraus eine einzelne komprimierte JavaScript-Datei erzeugt werden, die diese Pakete und Module mit ihren Abhängigkeiten enthält. So kann das Laden der gesamten notwendigen Ressourcen in einem Schritt und in komprimierter Form erfolgen, damit der Erstellungsprozess einer Dojo-Anwendung beschleunigt wird.

Beim AMD-Format werden Daten und Funktionen von einem Objekt in ein Modul gesetzt, sodass der ganze Quellcode für ein Modul auf einer einzigen Stelle steht. Dadurch ist der Quellcode leichter zu verstehen und man kann Fehler einfacher finden. Durch das schnelle und asynchrone Laden kann die Leistung der Web-Anwendungen verbessert werden. Dazu bietet das AMD-Format eine bessere Pakete-Portabilität und Debugg-Unterstützung.

Die wichtigen Komponenten von AMD sind die globalen Funktionen define() und require(). Die Erstellung eines Moduls erfolgt durch die Funktion define(), die in einer JavaScript-Datei abgelegt wird. Diese Funktion enthält normalerweise drei Parameter. Der erste Parameter ist optional und enthält eine ID für das Modul, damit dieses eindeutig identifiziert werden kann. Ohne diesen Parameter wird ein anonymes Modul erstellt. Der zweite Parameter ist ein Array mit einer Liste von Modulen, die die benötigten Ressourcen für das zu erstellende Modul umfassen und geladen werden müssen. Diese Module stellen die Abhängigkeit zum zu erstellenden Modul dar. Der dritte Parameter ist eine Callback-Funktion, die die geladenen Ressourcen bearbeitet und ausführt, sobald ihre modularen Abhängigkeiten verfügbar sind. In der folgenden Abbildung wird die Struktur der Funktion define() dargestellt.

```
define("modulID", ["modul1", "modul2", "modul3"], function (modul1, modul2, modul3) { … });
```

Abbildung 16: Struktur der Funktion define()

Die folgende Abbildung zeigt ein Beispiel von der Funktion define(), die in die JavaScript-Datei „begruessen.js“ gespeichert wird. Diese Datei befindet sich im benutzerdefinierte Paket test, somit lautet der Pfadname zum Modul test/begruessen. Dieses Modul benötigt das Modul dojo/dom, mit dem eine einfache Bearbeitung vom DOM-Objekt möglich ist. Dazu wird auch das Modul dom/Ready geladen, damit ein sicheres Laden des DOM-Objekts gewährleistet

wird. Die Callback-Funktion verwendet das DOM-Objekt und stellt die Unterfunktion setText zur Verfügung, um Texte in einem bestimmten Div-Tag einer Web-Anwendung auszugeben.

```
define(["dojo/dom", "dojo/domReady!"], function(dom){
   return {
      setText: function(id, text){
         var divTag = dom.byId(id);
         divTag.innerHTML = text;
      }
   };
});
```

Abbildung 17: Bespiel der Funktion define()

Nach der Erstellung des Moduls kann dieses beim Dojo-Loader registriert werden, indem das Paket test in das dojoConfig-Objekt der Web-Seite durch die Eigenschaft packages definiert wird.

```
<script>
         var dojoConfig = {
            async: true,
            packages: [{
               name: "test",
               location: "dojo-1.9.1/test"
            }]
         };
</script>
```

Abbildung 18: DojoConfig-Objekt mit der Definition des Pakets test

Nach der Registrierung des Moduls kann dieses vom Dojo-Loader durch den Aufruf von der Funktion require() geladen und verwendet werden. Die Funktion require() besitzt auch eine ähnliche Struktur wie die Funktion define(). Der erste Parameter ist auch ein Array mit einer Liste von Modulen, die mitgeladen werden. Bei require() werden aber nicht nur Module von Dojo geladen, sondern auch registrierte Module. Der zweite Parameter ist auch eine Callback-Funktion, die das Modul ausführt. Die Callback-Funktion kann aber auch von einem Ereignis der Web-Anwendung aufgerufen werden. Die folgende Abbildung zeigt die Struktur der Funktion require().

```
require(["modul1", "modul2", "modul3"], function (modul1, modul2, modul3) { … });
```

Abbildung 19: Struktur der Funktion require()

In der folgenden Abbildung wird ein Beispiel von einer Web-Anwendung gezeigt, in der die Funktion require() das erstellte und registrierte Modul test/begruessen verwendet. Im Quellcode des HTML-Dokuments wird zuerst das dojoConfig-Objekt erzeugt und dann der Dojo-Loader im asynchronen Modus geladen. Danach wird die Funktion require() aufgerufen und das Laden des Moduls aufgefordert. In der Callback-Funktion wird die Funktion setText des Moduls aufgerufen. Dabei werden die ID des Div-Tags, in dem ein Text ausgegeben werden soll, und der Text selbst als Parameter übergeben.

```
<!DOCTYPE HTML>
<html lang="de">
	<head>
		<script>
			var dojoConfig = {
				async: true,
				packages: [{
					name: "test",
					location: "dojo-1.9.1/test"
				}]
			};
		</script>

	</head>
	<body>
		<script src="dojo-1.9.1/dojo/dojo.js" ></script>

		<script>
			require(["test/begruessen"], function(begruessen){
				begruessen.setText("begruessung", "Hallo!");
			});
		</script>

		<div id="begruessung"></div>

	</body>
</html>
```

Abbildung 20: DojoConfig-Objekt mit der Definition des Pakets test

3. Konzeption

Der erste Schritt zur Erstellung des Konzepts für das Statistiktool war die Produktionsplattform (ADVBS06-Server) und die zugehörigen angewendeten Techniken zu analysieren, damit eine entsprechende Entwicklungsumgebung eingerichtet werden konnte. Diese gehören zur Vorgabe des Statistiktools.

Auf dem ADVBS06-Server, auf dem EDB-Portal läuft, ist der Server Apache Tomcat in der Version 6 installiert. Wie bei allen bereits bestehenden Trainern, werden die meisten Anwendungen des Portals in der Programmiersprache JSP implementiert. Zur Speicherung von Daten steht allen Trainern des EDB-Portals ein Oracle-Datenbanksystem zur Verfügung, in dem jeder Trainer auf sein eigenes Datenbankschema zugreift.

Das EDB-Portal bietet 16 Trainern zum Lernen über das Thema Datenbank an, die im Folgenden aufgelistet werden.

- 3NF-Trainer
- B-Baum-Zeichner
- DB-Kreuzworträtsel
- DB-Puzzle
- ER-Trainer
- JDBC-Trainer
- MCT (Multiple Choice Test)
- PL/SQL-Trainer
- REGEXP-Trainer
- SELECT2OBaum
- SQL-Abfrageoptimierungstool
- SQL-Sandbox
- SQL-Trainer
- SQL-Trainer2
- Stücklisten-Tool
- XQuery-Trainer

Zur Entwicklung des Statistiktools wurden alle Komponenten der Produktionsplattform in die Entwicklungsumgebung Eclipse Java EE IDE for Web Developers in der Version Juno (1.5) eingebunden. Diese Version von Eclipse enthält alle notwendigen Funktionen zur komfortablen Implementierung einer JSP-Anwendung.

Der nächste Schritt war, die Anforderungen an das Statistiktool festzulegen, damit die Eigenschaften und Funktionen, die das Statistiktool beinhalten soll, eingegrenzt werden konnten. Das Statistiktool soll auf Daten zugreifen, die bei der Verwendung von Trainern auf dem EDB-Portal gesammelt und in das Statistiktool-Schema gespeichert werden. Die Hauptanforderung an das Statistiktool ist, einen einfachen und schnellen Überblick über diese Daten zu ermöglichen. Die Bestimmung aller Eigenschaften des Statistiktools konnte erst nach einer Analyse des Statistiktool-Schemas und der gesammelten Daten erfolgen. Mit der Auswertung der daraus entstandenen Erkenntnisse wurde dann bestimmt, welche Informationen aus den Daten der Tabellen wichtig sind und mit diesen Informationen Statistiken erstellt werden können.

Die Analyse ergab auch, dass eine übersichtliche Darstellung von einer so großen Menge an Daten nur als Diagramme möglich ist. Aus diesem Grund musste ein zusätzliches Framework ausgewählt und in die Implementierung des Statistiktools hinzugefügt werden, damit die Erstellung von Diagrammen einfacher gestaltet werden konnte.

In den folgenden Kapiteln werden die einzelnen Schritte des Konzepts mit den Anforderungen an das Statistiktool, der Analyse des Statistiktool-Schemas, der Erkenntnisse der Analyse und der Auswahl des DOJO Toolkit beschrieben.

3.1 Anforderungen an das Statistiktool

Das Statistiktool hat einige festgelegte Anforderungen zu erfüllen, die im Folgenden aufgelistet werden:

- Einstellung der Benutzerabfrage
- Stand der Statistik anzeigen
- Einen bestimmbaren Zeitraum zur Datenabfrage
- Daten aus verschiedenen Hochschulen getrennt anzeigen
- einfache und übersichtliche Darstellung der Daten
- schnelle Anzeige der Daten
- Keine Datenveränderung ermöglichen

Die erste Anforderung lautet, dass die Benutzer im Statistiktool die Datenabfrage zur Anzeige der Statistik selbst einstellen können. Dafür enthält das Statistiktool Formulare, auf die die Benutzer bestimmte Auswahlen zur Datenabfrage treffen können.

Das Statistiktool soll auch eine Anzeige besitzen, die den Stand der Statistik darstellen soll. Die gesammelten Daten werden in das Statistiktool-Schema mit dem Datum der Datenerfassung gespeichert. Mit dem minimalen und maximalen Wert dieses Datums kann man bestimmen, seit wann und bis wann Daten gesammelt wurden und im Statistiktool-Schema vorhanden sind. Diese Informationen stellen den Stand der Statistik dar und werden auf die Anzeige im Statistiktool ausgegeben.

Diese Eigenschaft führt auch dazu, dass ein Zeitraum bei der Einstellung der Datenabfrage vom Benutzer ausgewählt werden kann. Damit kann der Benutzer festlegen, dass Statistiken nur von einem bestimmten Zeitraum im Statistiktool angezeigt werden. Auf dem Formular der Datenabfrage kann der Zeitraum eingestellt werden und besteht aus den Elementen Tag, Monat und Jahr.

Auf dem EDB-Portal können Studenten aus unterschiedlichen Hochschulen zugreifen und dessen Trainer verwenden. Die gesammelten Daten werden in das Statistiktool-Schema gespeichert, ohne dass eine Unterscheidung der Daten nach Hochschulen gemacht wird. Damit

können die Daten nach Hochschulen nicht ohne die Verwendung weiterer Quellen getrennt werden. Der Vorteil einer solchen Trennung von Daten liegt darin, dass die Professoren die Leistungen von ihren eigenen Studenten überprüfen könnten.

Die wichtige Anforderung ist, die Daten in einer einfachen und übersichtlichen Art darzustellen. Dies wird erreicht, indem die Daten als Diagramme gezeigt werden, damit die Informationen schneller erfasst und verglichen werden können. Wenn zu viele Daten in die Diagramme angezeigt werden sollen, soll der Benutzer die Möglichkeit haben, die Anzahl der Daten einzugrenzen. Somit soll sichergestellt werden, dass die Diagramme übersichtlich bleiben. Dazu kommt auch, dass die Daten schnell angezeigt werden sollen. Dafür wird die Technik AJAX eingesetzt, damit die Web-Seite des Statistiktools nicht immer wieder neu geladen werden muss, wenn neue Daten abgefragt und auf die Seite eingefügt werden.

Das Statistiktool soll auch keine Möglichkeit zur Datenveränderung erlauben. Sonst können die gesammelten Daten verfälscht oder sogar gelöscht werden.

3.2 Analyse des Statistiktool-Schemas

Die Analyse des Statistiktool-Schemas soll nicht nur Erkenntnisse über die Struktur der Tabellen liefern, sondern auch, welche Daten in den Tabellen wichtige Informationen für das Statistiktool enthalten und wie groß diese Tabellen sind.

Das Statistiktool-Schema enthält insgesamt 18 Tabellen. Dazu besitzt das Schema auch andere Komponenten bzw. Elementen, wie Trigger und Sequenzen, die aber für das Statistiktool nicht relevant sind.

Zwischen den Tabellen gibt es keine besondere Verbindung, weil in die meisten Tabellen Daten aus unterschiedlichen Quellen (Trainer des EDB-Portals) gespeichert werden. Daraus folgt, dass die Tabellen getrennt voneinander analysiert werden. Bei der Analyse der Tabellen konnte man durch die Kardinalität schon erkennen, dass einige Trainer von Benutzern nicht oft verwendet werden.

3.2.1 Inhalt und Beschreibung der Tabellen

Aus den 18 Tabellen des Statistiktool-Schemas wurden einige Tabellen von EDB-Administratoren erstellt und enthalten Informationen über die Trainer des EDB-Portals. Die meisten Tabellen wurden aber von der Studenten-Gruppe angelegt, die zur Realisierung des Statistiktool-Schemas zuständig war und enthalten Daten, die bei der Verwendung der Trainer durch die Benutzer gesammelt werden.

Im Folgenden werden einzelne Tabellen des Statistiktool-Schemas ausführlich beschrieben.

3.2.1.1 Tabelle BENUTZER

Die Tabelle BENUTZER wurde von EDB-Administratoren angelegt und enthält Einloggen-Daten, die zur Anmeldung bei dem Frontend der Studenten-Gruppe dient. Die Tabelle besteht aus den Spalten ID (Primärschlüssel), NAME, die den Benutzername enthält, und PASSWORT, in die das zugehörige Passwort gespeichert wird. Für die Erstellung des Statistiktools werden diese Daten nicht benötigt.

```
BENUTZER (
        ID              NUMBER,
        NAME            VARCHAR2(20),
        PASSWORT        VARCHAR2(20)
);
```

Abbildung 21: Struktur der Tabelle BENUTZER

ID	NAME	PASSWORT
2	admin	admin

Abbildung 22: Inhalt der Tabelle BENUTZER

(Stand: 04.05.2013, 23:30 Uhr)

3.1.1.2 Tabelle NATION

Die Tabelle NATION wurde von EDB-Administratoren erstellt. Die Tabelle besteht aus den Spalten ID (Primärschlüssel) und NATIONNAME, die die von dem EDB-Portal unterstützten Sprachen enthält. Nicht alle Trainer unterstützen verschiedene Sprachen. Zurzeit sind nur die Sprachen Deutsch und Englisch in der Tabelle eingetragen.

```
NATION (
        ID                  NUMBER,
        NATIONNAME          VARCHAR2(150)
);
```

Abbildung 23: Struktur der Tabelle NATION

ID	NATIONNAME
1	deutsch
2	englisch

Abbildung 24: Inhalt der Tabelle NATION

(Stand: 04.05.2013, 23:30 Uhr)

3.1.1.3 Tabelle NATION_AUFGERUFEN

Die Tabelle NATION_AUFGERUFEN wurde von der Studenten-Gruppe erstellt und besteht aus den Spalten DATUM und NATION_ID, anhand derer festgelegt werden kann, wann eine bestimmte Sprache von einem Benutzer ausgewählt wurde.

```
NATION_AUFGERUFEN (
        DATUM               DATE,
        NATION_ID           NUMBER
);
```

Abbildung 25: Struktur der Tabelle NATION_AUFGERUFEN

DATUM	NATION_ID
03.04.12	1
03.04.12	1
03.04.12	1
03.04.12	2
02.04.12	1
03.04.12	1
03.04.12	1
03.04.12	2
03.04.12	2
04.04.12	2
04.04.12	1
04.04.12	1
04.04.12	1
04.04.12	2
04.04.12	1

Abbildung 26: Teilinhalt der Tabelle NATION_AUFGERUFEN

(Stand: 04.05.2013, 23:30 Uhr)

3.1.1.4 Tabelle TOOLS

Die Tabelle TOOLS mit den Spalten ID (Primärschlüssel) und NAME wurde von EDB-Administratoren erstellt und enthält die Namen aller Trainer bzw. Tools, die auf dem EDB-Portal zur Verfügung stehen. Zurzeit werden 16 Trainer vom EDB-Portal angeboten und sind in dieser Tabelle eingetragen, wie in der Abbildung 28 **„**Inhalt der Tabelle TOOLS“ gezeigt wird.

```
TOOLS (
        ID              NUMBER,
        NAME            VARCHAR2(50)
);
```

Abbildung 27: Struktur der Tabelle TOOLS

ID	NAME
2	MCT
3	ER-Trainer
4	SQL-Trainer
5	SQL-Trainer2
6	SQL-Sandbox
7	JDBC-Trainer
8	PL/SQL-Trainer
9	3NF-Trainer
10	SELECT2OBaum
11	DB-Puzzle
12	DB-Kreuzworträtsel
13	XQuery-Trainer
14	B-Baum-Zeichner
15	REGEXP-Trainer
16	Stücklisten-Tool
17	SQL-Abfrageoptimierungstool

Abbildung 28: Inhalt der Tabelle TOOLS

(Stand: 04.05.2013, 23:30 Uhr)

3.1.1.5 Tabelle TOOL_AUFGERUFEN

Die Tabelle TOOL_AUFGERUFEN erfasst die Aufrufe aller Trainer bzw. Tools auf dem EDB-Portal, die durch die Benutzer erfolgen. In die Spalte TOOL_ID wird die ID des Trainers aus der Tabelle TOOLS gespeichert. In die Spalte DATUM wird der genaue Zeitpunkt des Aufrufes eingetragen.

```
TOOL_AUFGERUFEN (
        DATUM           DATE,
        TOOL_ID         NUMBER
);
```

Abbildung 29: Struktur der Tabelle TOOL_AUFGERUFEN

DATUM	TOOL_ID
03.04.12	4
03.04.12	4
03.04.12	4
03.04.12	4
04.04.12	5
04.04.12	4
04.04.12	4
04.04.12	4
04.04.12	4
04.04.12	6
04.04.12	6
04.04.12	6
04.04.12	6
04.04.12	6
04.04.12	6

Abbildung 30: Teilinhalt der Tabelle TOOL_AUFGERUFEN

(Stand: 04.05.2013, 23:30 Uhr)

3.1.1.6 Tabelle TOOL_BEENDET

In die Tabelle TOOL_BEENDET wird eine neue Zeile eingetragen, wenn ein Benutzer einen Trainer bis zum Ende verwendet hat. Mit den Spalten TOOL_ID und DATUM wird die ID des Trainers aus der Tabelle TOOLS und den genauen Zeitpunkt des Beendens gespeichert. Das Beenden eines Trainers kann nur bei dem MCT, SQL-Trainer, SQL-Trainer2 und XQuery bestimmt werden.

```
TOOL_BEENDET (
        DATUM               DATE,
        TOOL_ID             NUMBER
);
```

Abbildung 31: Struktur der Tabelle TOOL_BEENDET

DATUM	TOOL_ID
04.04.12	4
04.04.12	4
04.04.12	4
04.04.12	4
04.04.12	4
05.04.12	4
06.04.12	4
03.04.12	4
04.04.12	4
06.04.12	4
06.04.12	4
09.04.12	4
09.04.12	4
10.04.12	2
11.04.12	2

Abbildung 32: Teilinhalt der Tabelle TOOL_ BEENDET

(Stand: 04.05.2013, 23:30 Uhr)

3.1.1.7 Tabelle SQL_OPTIMIZER_SCHEMA

In die Tabelle SQL_OPTIMIZER_SCHEMA werden Daten über die Verwendung des SQL-Abfrageoptimierungstools gespeichert. Bei dem SQL-Abfrageoptimierungstool wird der Ausführungsplan[9] einer SELECT-Anweisung angezeigt, die vom Benutzer ausgewählt oder selbst geschrieben werden kann. Die IDs der ausgewählten SELECT-Anweisung werden in die Spalte AUFGABE_ID und das entsprechende Schema in die Spalten SCHEMA_ID eingetragen. Bei selbst geschriebenen SELECT-Anweisung wird der Wert -1 in die Spalten AUFGABE_ID gespeichert. Dazu wird auch der Zeitpunkt des Toolaufrufs in die Spalte DATUM gespeichert.

[9] *Kapitel „3.3.4.1 Annahme eines Performance-Problems“*

```
SQL_OPTIMIZER_SCHEMA (
        DATUM               DATE,
        SCHEMA_ID           NUMBER,
        AUFGABE_ID          NUMBER
);
```

Abbildung 33: Struktur der Tabelle SQL_OPTIMIZER_SCHEMA

DATUM	SCHEMA_ID	AUFGABE_ID
20.01.13	2	43
24.01.13	2	64
24.01.13	2	67
26.01.13	1	1
26.01.13	1	1
26.01.13	1	-1
28.01.13	1	3
28.01.13	2	74
31.01.13	1	38
31.01.13	1	6
31.01.13	1	33
31.01.13	1	38
07.03.13	1	1
07.03.13	2	43
07.03.13	2	43

Abbildung 34: Teilinhalt der Tabelle SQL_OPTIMIZER_SCHEMA

(Stand: 04.05.2013, 23:30 Uhr)

3.1.1.8 Tabelle SQL_TRAINER_1_SCHEMA

Die Tabelle SQL_TRAINER_1_SCHEMA beinhaltet Daten über die Verwendung des SQL-Trainers. Bei dem SQL-Trainer wird eine Aufgabenstellung ausgegeben und der Benutzer muss die entsprechende SELECT-Anweisung in einem Textfeld eingeben. Zu dieser Tabelle gehören die Spalten AUFGABE_ID und SCHEMA_ID, die die Informationen über die Aufgabenstellung speichern. Die Spalte SCHEMA_ID hat den Datentyp VARCHAR2, weil der Name des Datenbankschemas und nicht die tatsächliche ID des Schemas gespeichert wird. Zu dieser Tabelle gehört auch die Spalte DATUM, in die der Zeitpunkt des Traineraufrufs einge-

tragen wird. Es werden auch Daten über die Ergebnisse der Aufgaben in die Tabelle FRAGENAUWERTUNG gespeichert.

```
SQL_TRAINER_1_SCHEMA (
        DATUM               DATE,
        SCHEMA_ID           VARCHAR2(100),
        AUFGABE_ID          NUMBER
);
```

Abbildung 35: Struktur der Tabelle SQL_TRAINER_1_SCHEMA

DATUM	SCHEMA_ID	AUFGABE_ID
03.04.12	fahrrad	61
03.04.12	fahrrad	88
02.04.12	busse	214
03.04.12	fahrrad	16
03.04.12	fahrrad	64
03.04.12	fahrrad	50
03.04.12	fahrrad	48
04.04.12	fahrrad	48
04.04.12	fahrrad	89
04.04.12	fahrrad	83
04.04.12	fahrrad	51
04.04.12	fahrrad	87
04.04.12	fahrrad	89
04.04.12	fahrrad	70
04.04.12	fahrrad	61
04.04.12	busse	197
04.04.12	busse	197

Abbildung 36: Teilinhalt der Tabelle SQL_TRAINER_1_SCHEMA

(Stand: 04.05.2013, 23:30 Uhr)

3.1.1.9 Tabelle SQL_TRAINER_SCHEMA

In der Tabelle SQL_TRAINER_SCHEMA werden Daten zum SQL-Trainer2 gespeichert. Der SQL-Trainer2 besitzt die gleichen Funktionalitäten des SQL-Trainers. Bei dem SQL-Trainer2 wird eine Aufgabenstellung ausgegeben und der Benutzer muss die entsprechende SELECT-

Anweisung in einem Textfeld eingeben. Die Tabelle besitzt auch die Spalten AUFGABE_ID und SCHEMA_ID, in die die Informationen über die Aufgabenstellung gespeichert werden. Dazu gibt es die Spalte DATUM, in die der Zeitpunkt des Traineraufrufs eingetragen wird. Es werden auch Daten über die Ergebnisse der Aufgaben in die Tabelle FRAGENAUWERTUNG gespeichert.

```
SQL_TRAINER_SCHEMA (
        DATUM               DATE,
        SCHEMA_ID           NUMBER,
        AUFGABE_ID          NUMBER
);
```

Abbildung 37: Struktur der Tabelle SQL_TRAINER_SCHEMA

DATUM	SCHEMA_ID	AUFGABE_ID
09.04.12	1	903
09.04.12	1	903
15.04.12	6	864
04.04.12	8	1162
09.04.12	1	903
15.04.12	6	886
15.04.12	1	903
15.04.12	1	853
24.04.12	1	924
24.04.12	1	924
29.04.12	6	862
29.04.12	6	862
29.04.12	6	957
29.04.12	6	957
29.04.12	6	957

Abbildung 38: Teilinhalt der Tabelle SQL_TRAINER_SCHEMA

(Stand: 04.05.2013, 23:30 Uhr)

3.1.1.10 Tabelle XQUERY_SCHEMA

In die Tabelle XQUERY_SCHEMA werden Informationen zum XQuery-Trainer eingetragen. Bei dem XQuery-Trainer soll ein FLOWR-Ausdruck auf ein XML-Dokument angewendet

werden, indem eine Aufgabenstellung ausgegeben wird und der Benutzer den entsprechenden FLOWR-Ausdruck in einen Text eingibt. Die Tabelle hat die Spalten AUFGABE_ID und SCHEMA_ID, die die Informationen über die Aufgabenstellung enthalten. Die Spalte SCHEMA_ID ist vom Datentyp VARCHAR2 und nicht vom Datentyp NUMBER, was üblich wäre. Zu der Tabelle gehört auch die Spalte DATUM, in die der Zeitpunkt des Traineraufrufs eingetragen wird. Es werden auch Daten über die Ergebnisse der Aufgaben in die Tabelle FRAGENAUWERTUNG gespeichert.

```
XQUERY_SCHEMA (
        DATUM           DATE,
        SCHEMA_ID       VARCHAR2(15),
        AUFGABE_ID      NUMBER
);
```

Abbildung 39: Struktur der Tabelle XQUERY_SCHEMA

DATUM	SCHEMA_ID	AUFGABE_ID
12.05.12	1	99
19.05.12	1	56
19.05.12	1	56
19.05.12	1	56
19.05.12	1	56
26.06.12	1	54
26.06.12	1	7
26.06.12	1	7
26.06.12	1	7
26.06.12	1	7
26.06.12	1	7
26.06.12	1	52
26.06.12	1	99
26.06.12	1	99
26.06.12	2	19
26.06.12	2	15
26.06.12	2	20

Abbildung 40: Teilinhalt der Tabelle XQUERY_SCHEMA

(Stand: 04.05.2013, 23:30 Uhr)

3.1.1.11 Tabelle MCT_UNI

Die Tabelle MCT_UNI mit den Spalten ID (Primärschlüssel) und NAME wurde von EDB-Administratoren angelegt und enthält alle Hochschulen (Universitäten und Fachhochschulen), die auf das EDB-Portal zugreifen können. Bei dem MCT handelt es sich um einen Multiple-Choice-Test. In diesem Test werden 20 Fragen ausgegeben und die Benutzer müssen entweder die richtige Antwort in einen Textfeld eintippen oder sie ankreuzen.

```
MCT_UNI (
        ID                  NUMBER,
        UNI_NAME            VARCHAR2(50)
);
```

Abbildung 41: Struktur der Tabelle MCT_UNI

ID	UNI_NAME
14	DBS FH Köln
15	DuI Verbundstudium Dortmund/Köln
16	DBS FH Düsseldorf
17	DBS FH Hannover
18	DBS Uni Duisburg/Essen (Auszug)
19	DBS FH Brandenburg
20	DBS FHW-Berlin
21	DBS Beuth Hochschule Berlin
22	DBS Uni Duisburg/Essen (prüfungsrelevant)
23	FH TEST
24	FH Köln aktuell

Abbildung 42: Inhalt der Tabelle MCT_UNI

(Stand: 04.05.2013, 23:30 Uhr)

3.1.1.12 Tabelle MCT_UNI_AUSGEWAEHLT

Bevor der Trainer MCT auf dem EDB-Portal gestartet werden kann, muss der Benutzer eine Hochschule auswählen. In die Tabelle MCT_UNI_AUSGEWAEHLT wird bei jeder Auswahl einer Hochschule eine neue Zeile eingetragen. Die Struktur dieser Tabelle wurde von EDB-

Administratoren festgelegt. Die Werte der Spalten MCT_UNI_ID entsprechen den Werten von der Spalte ID aus der Tabelle MCT_UNI. Dazu werden auch in die Spalten SESSION_ID und DATUM die Session sowie der genaue Zeitpunkt der Auswahl gespeichert.

```
MCT_UNI_AUSGEWAEHLT (
        DATUM               DATE,
        MCT_UNI_ID          NUMBER,
        SESSION_ID          VARCHAR2(32)
);
```

Abbildung 43: Struktur der Tabelle MCT_UNI_AUSGEWAEHLT

DATUM	MCT_UNI_ID	SESSION_ID
02.04.12	22	7222B348F0CAB55BF4BD9EFD76EE3BC6
03.04.12	14	10227FF2AE8084287C6CAA99AD84A0AA
03.04.12	15	57313161E2FC724BC9C1918E22BBCD26
04.04.12	14	3357261A80E7D182DA300DB6FD370432
04.04.12	15	D1E55F43D90F7C2CA2D272D973FC4A96
05.04.12	15	0B4E8C0C644B93EDDE7EBFCB686133DE
07.04.12	14	873259C3508498DBE3BE4E23C356C6F8
07.04.12	14	43E43C1F8531149D136ADD7C5D075C69
09.04.12	14	AB26EBB9484A947D0E3C9F6F50336506
09.04.12	14	3B6718BB908722297BFF0C75A3153CB5
10.04.12	14	9AC49E1A98A87D8B0FC82C2DBA2ACA76
11.04.12	14	B2D0480C54A3A9B0DA6FE7C90A12D26C
11.04.12	14	B9EA50434BA179E15EC26D6412B9E929
11.04.12	14	DC6B0859F502019497BA3C1F34A05627
11.04.12	22	5758198562CA13F9D51F6ADCAEACA6AB
11.04.12	15	FE384D9EA853219941028DEA8D6D95B2

Abbildung 44: Teilinhalt der Tabelle MCT_UNI_AUSGEWAEHLT

(Stand: 04.05.2013, 23:30 Uhr)

3.1.1.13 Tabelle MCT_TIME

Die Tabelle MCT_TIME ist nur eine Hilfstabelle. Die Spalten SESSION_ID und STARTTIME unterstützen die Berechnung der benötigten Antwortzeit für eine Frage im MCT. Nach-

dem diese Daten verwendet wurden, werden sie gelöscht. Die Daten aus dieser Tabelle sind für die Erstellung des Statistiktools nicht relevant.

```
MCT_TIME (
        SESSION_ID          VARCHAR2(32),
        STARTTIME           NUMBER
);
```

Abbildung 45: Struktur der Tabelle MCT_TIME

SESSION_ID	STARTTIME
988B98525465F910B1A438D05B972BE3	1354010635025
82CC3FBC24A751BB84001A79E84C641D	1358892559058
7916C9BE977142922CDFCC0C7325D45B	1354009220272
967F201FCA5A8F3CDAEADEB2D16F2F1A	1355301730061
6F455C73B9607AF6B2A0C4D8BCCE3952	1354010739956
F919E60B4C742FDFCA78DB6CF693CFE5	1360663312236
25BAE607480DEC4A8DEC3A7A47E27D7A	1354009264386
CB301D4E13CB51B82088A8BC48C02D1C	1359217379776
89E25D8CB5B88AF8182E43DBBB5BE912	1360694895292
D419E7315BA26E9CDB6CCEBA632BCF59	1362399278197
85E83A94C22191D50224D810D6834D6E	1360697405159
EE11CDC478B2A1485A74DAEB8CC6521B	1360761204147
EE11CDC478B2A1485A74DAEB8CC6521B	1360761193313
EE11CDC478B2A1485A74DAEB8CC6521B	1360761220444
0509C833F2C943536EF16C6ED7C96863	1362402557869
1E49D53F36912DF935C8A53772BD0849	1362680388254

Abbildung 46: Teilinhalt der Tabelle MCT_TIME

(Stand: 04.05.2013, 23:30 Uhr)

3.1.1.14 Tabelle MCT_KATEGORIEN

Die Tabelle MCT_KATEGORIEN wurde von EDB-Administratoren erstellt und enthält alle Themenkategorien des MCTs, die von Benutzern ausgewählt werden können. Die Tabelle besteht aus den Spalten ID (Primärschlüssel), UNI_ID, KATNR und BEZEICHNUNG. Die Werte aus der Spalte UNI_ID kommen aus der Tabelle MCT_UNI. Es ist wichtig, zu beach-

ten, dass die Universitäten und Fachhochschulen für die gleiche Kategorie unterschiedliche Bezeichnungen verwenden.

```
MCT_KATEGORIEN (
        ID                  NUMBER,
        UNI_ID              NUMBER
        KATNR               NUMBER
        BEZEICHNUNG         VARCHAR2(100)
);
```

Abbildung 47: Struktur der Tabelle MCT_KATEGORIEN

ID	UNI_ID	KATNR	BEZEICHNUNG
1	14	1	K1: Einführung in die Grundbegriffe der Datenbanken
2	14	2	K2: Ein Phasenmodell der Datenbankentwicklung
3	14	3	K3: Das ER-Modell und das EERM (erweitertes Entity-Relationship-Modell)
4	14	4	K4: Datenmodelle, relationale Algebra und Anfrageverarbeitung
5	14	5	K4: Funktionale Abhängigkeiten, Normalformen und Transformation auf ein relationales Datenmodell
6	14	6	K5: Die Datenbanksprache SQL2003: relationale Bestandteile (DDL)
7	14	7	K5: Die Datenbanksprache SQL2003: relationale Bestandteile (DML, DAL und DQL)
8	14	8	K6: Objektrelationales SQL2003
9	14	10	K6: Objektrelationale Anwendungsprogrammierung: SQLJ
10	14	12	K7: Anwendungsprogrammierung: Aktive DBMS und PL/SQL-Trigger
11	14	13	K8: Transaktionen und verwandte Konzepte
12	14	14	K9: Physische Speicherstrukturen
13	18	1	K1: Einführung in die Grundbegriffe der Datenbanken
14	18	2	K2: Ein Phasenmodell der Datenbankentwicklung
15	18	3	K3: Das ER-Modell und das EERM (erweitertes Entity-Relationship-Modell)
16	18	4	K4: Datenmodelle, relationale Algebra und Anfrageverarbeitung
17	18	5	K4: Funktionale Abhängigkeiten, Normalformen und Transformation auf ein relationales Datenmodell
18	18	6	K5: Die Datenbanksprache SQL2003: relationale Bestandteile (DDL)
19	18	7	K5: Die Datenbanksprache SQL2003: relationale Bestandteile (DML, DAL und DQL)
20	18	8	K6: Objektrelationales SQL2003
21	18	10	K6: Objektrelationale Anwendungsprogrammierung: SQLJ
22	18	12	K7: Anwendungsprogrammierung: Aktive DBMS und PL/SQL-Trigger
23	18	13	K8: Transaktionen und verwandte Konzepte
24	18	14	K9: Physische Speicherstrukturen
25	18	11	K7: Anwendungsprogrammierung: Oracle PL/SQL (Prozeduren und Funktionen)
26	18	9	K6: Objektrelationale Anwendungsprogrammierung: JDBC
27	21	1	K1: Einführung in die Grundbegriffe der Datenbanken
28	21	2	K2: Ein Phasenmodell der Datenbankentwicklung
29	21	3	K3: Das ER-Modell und das EERM (erweitertes Entity-Relationship-Modell)
30	21	4	K4: Datenmodelle, relationale Algebra und Anfrageverarbeitung
31	21	5	K4: Funktionale Abhängigkeiten, Normalformen und Transformation auf ein relationales Datenmodell
32	21	6	K5: Die Datenbanksprache SQL2003: relationale Bestandteile (DDL)
33	21	7	K5: Die Datenbanksprache SQL2003: relationale Bestandteile (DML, DAL und DQL)
34	21	8	K6: Objektrelationales SQL2003

Abbildung 48: Teilinhalt der Tabelle MCT_KATEGORIEN

(Stand: 04.05.2013, 23:30 Uhr)

3.1.1.15 Tabelle MCT_KATEGORIEN_AUSGEWAEHLT

In die Tabelle MCT_KATEGORIEN_AUSGEWAEHLT werden Daten über die Auswahl der Kategorien im MCT eingetragen. Diese Auswahl wird vor dem Start des MCTs von jedem Benutzer getroffen. Die Tabelle enthält die Spalten DATUM und MCT_KAT_ID. Die Werte der Spalte MCT_KAT_ID entsprechen den Werten der Spalte KATNR aus der Tabelle MCT_KATEGORIEN. In die Spalte DATUM wird der genaue Zeitpunkt der Kategorienauswahl gespeichert.

```
MCT_KATEGORIEN_AUSGEWAEHLT (
        DATUM               DATE,
        MCT_KAT_ID          NUMBER
);
```

Abbildung 49: Struktur der Tabelle MCT_KATEGORIEN_AUSGEWAEHLT

DATUM	MCT_KAT_ID
11.04.12	1
11.04.12	2
11.04.12	3
11.04.12	4
12.04.12	1
12.04.12	3
12.04.12	4
12.04.12	5
12.04.12	6
12.04.12	7
12.04.12	9
12.04.12	1
12.04.12	1
05.04.12	1
05.04.12	3

Abbildung 50: Teilinhalt der Tabelle MCT_KATEGORIEN_AUSGEWAEHLT

(Stand: 04.05.2013, 23:30 Uhr)

3.1.1.16 Tabelle MCT_FEEDBACK

Die Tabelle MCT_FEEDBACK enthält Daten zu den hinterlassenden Feedbacks im MCT und besteht aus den Spalten DATUM und FRAGE_ID. In diesen Spalten wird die ID der Frage mit dem Zeitpunkt der Feedback-Hinterlassung gespeichert.

```
MCT_FEEDBACK (
        DATUM               DATE,
        FRAGE_ID            NUMBER
);
```

Abbildung 51: Struktur der Tabelle MCT_FEEDBACK

DATUM	FRAGE_ID
09.05.12	92468
18.04.12	20400
02.05.12	2700
10.05.12	1978
15.05.12	92451
15.05.12	1492
02.06.12	41693
17.06.12	2360
18.06.12	2375
20.06.12	2294
21.06.12	2360
24.06.12	22025
25.06.12	22451
26.06.12	22758
02.07.12	2754

Abbildung 52: Teilinhalt der Tabelle MCT_FEEDBACK

(Stand: 04.05.2013, 23:30 Uhr)

3.1.1.17 Tabelle MCT_HILFE

Die Tabelle MCT_HILFE wird immer dann mit Datensätzen gefüllt, wenn ein Benutzer die Hilfe bei einer Frage im MCT aufruft. Die Tabelle enthält die Spalten FRAGE_ID und DATUM, in die die ID der Frage mit dem Zeitpunkt des Aufrufes gespeichert werden.

```
MCT_HILFE (
        DATUM               DATE,
        FRAGE_ID            NUMBER
);
```

Abbildung 53: Struktur der Tabelle MCT_HILFE

DATUM	FRAGE_ID
10.04.12	414
10.04.12	2007
17.04.12	51411
17.04.12	2095
17.04.12	525
17.04.12	1154
18.04.12	91075
18.04.12	91075
18.04.12	92100
18.04.12	90414
05.04.12	22479
10.04.12	1791
18.04.12	92094
18.04.12	92451
18.04.12	91795

Abbildung 54: Teilinhalt der Tabelle MCT_HILFE

(Stand: 04.05.2013, 23:30 Uhr)

3.1.1.18 Tabelle FRAGENAUSWERTUNG

Die Tabelle FRAGENAUSWERTUNG enthält die Ergebnisse der von Benutzern durchgeführten Aufgaben. Die Tabelle besteht aus den Spalten ID (Primärschlüssel), TOOL_ID, FRAGE_ID, RICHTIG_BEANTWORTET, ZEIT und DATUM. Die Spalte ID dient zur eindeutigen Identifikation der Datensätzen. Durch den Trigger INSERT_ID_FRAGE mit der Sequence COUNT_ID_FRAGE werden die Werte in diese Spalte eingefügt. Die Werte der Spalte TOOL_ID entsprechen den Werten der Spalte ID aus der Tabelle TOOLS. Die Überprüfung auf richtige oder falsche Antwort ist nur bei dem MCT, SQL-Trainer, SQL-Trainer2 und XQuery-Trainer möglich. Somit kann die Spalte TOOL_ID nur die Werte der Trainer-IDs 2, 4, 5 und 13 aufnehmen. Mit der Spalte FRAGE_ID kann die Frage bzw. Aufgabe des Trainers identifiziert werden. In der Spalte RICHTIG_BEANTWORTET können nur die Werte 0 oder 1 gespeichert werden. Der Wert 0 steht für falsch beantwortet und der Wert 1 für richtig beantwortet. In der Spalte ZEIT wird die benötigte Zeit für die Beantwortung der Frage im MCT gespeichert. Bei dem SQL-Trainer, SQL-Trainer 2 und XQuery-Trainer kann keine Zeit berechnet werden und aus diesem Grund ist der Wert der Spalte bei diesen Trainern immer 0. Bei der Berechnung der benötigten Zeit wird die Hilfstabelle MCT_TIME verwendet. In die Spalte DATUM wird der Zeitpunkt der Fragebeantwortung eingetragen. Abschließend ist noch zu erwähnen, dass die Tabelle FRAGENAUSWERTUNG die größte Tabelle des Statistiktool-Schemas ist.

```
FRAGENAUSWERTUNG (
        ID                          NUMBER,
        TOOL_ID                     NUMBER,
        FRAGE_ID                    NUMBER,
        RICHTIG_BEANTWORTET         NUMBER,
        ZEIT                        NUMBER,
        DATUM                       DATE
);
```

Abbildung 55: Struktur der Tabelle FRAGENAUSWERTUNG

ID	TOOL_ID	FRAGE_ID	RICHTIG_BEANTWORTET	ZEIT	DATUM
2205	5	862	0	0	29.04.12
2206	5	892	0	0	29.04.12
2207	4	127	0	0	29.04.12
2208	4	131	0	0	29.04.12
2209	4	155	0	0	29.04.12
2210	4	100	0	0	29.04.12
2211	2	20406	1	20,83399987220764	30.04.12
2212	2	20632	1	32,176000118255615	30.04.12
2213	2	22010	0	25,361999988555908	30.04.12
2214	2	22016	1	13,57200002670288	30.04.12
2215	2	22572	0	12,974000215530396	30.04.12
2216	2	22594	1	91,21099996566772	30.04.12
2217	2	21126	0	7,152000188827515	30.04.12
2218	2	22562	0	26,19599986076355	30.04.12
2219	2	22597	1	211,79899978637695	30.04.12
2220	2	22564	0	173,5899999141693	30.04.12
2221	2	22813	0	23,20199990272522	30.04.12
2222	2	22581	0	16,407999992370605	30.04.12

Abbildung 56: Teilinhalt der Tabelle FRAGENAUSWERTUNG

(Stand: 04.05.2013, 23:30 Uhr)

3.2.2 Indizes und Primärschlüssel der Tabellen

Zur Analyse des Statistiktool-Schemas gehört die Untersuchung von Indizes und Primärschlüsseln der Tabellen. Die Gründe dafür sind, dass indizierte Daten schneller aufgerufen werden können, und dass die Primärschlüssel in die WHERE-Klausel von SELECT-Anweisungen im Statistiktool verwendet werden müssen. Bei der Erstellung von Primärschlüsseln wird auch ein Index für die Schlüsselspalten automatisch angelegt.

Die meisten Tabellen des Statistiktool-Schemas, die von der Studenten-Gruppe erstellt wurden, besitzen keinen Primärschlüssel und keinen Index. Die einzige Ausnahme ist die Tabelle FRAGENAUSWERTUNG, die ein Primärschlüssel enthält.

Die anderen Tabellen, für die Primärschlüssel erstellt wurden, wurden von EDB-Administratoren angelegt und enthalten keine gesammelten Daten, sondern bestimmte Infor-

mationen über die Trainer und Tools, wie z. B. die Tabelle TOOLS, die die IDs und die Namen der vom EDB-Portal angebotenen Tools beinhaltet.

Der Vorteil von Indizes liegt darin, dass sie den Zugriff auf Daten beschleunigen. Die Indizes einer Tabelle stellen eine eigene sortierte Speicherstruktur mit den Indexwerten und die Speicheradressen der Daten dar. Wegen ihrer geringen Datenlänge können Indizes mit weniger Leseoperationen als die Daten selbst in den Hauptspeicher gespeichert werden. Der Nachteil von Indizes besteht darin, dass sie das Schreiben von Daten verlangsamen. So wird das Speichern von Datensätzen in Spalten ohne Index schneller als in indizierte Spalten ausgeführt, weil bei einer INSERT-Anweisung ein neuer Indexeintrag erzeugt wird und alle Indizes aktualisiert werden müssen. Diese Aktualisierung wird Index-Rebuild genannt.

Der Grund für das Fehlen von Indizes und Primärschlüsseln liegt wahrscheinlich darin, dass die Tabellen nur für das Schreiben von Daten verwendet werden sollen. Eine Abfrage der Daten, wie im Statistiktool gefordert ist, war damals noch nicht vorgesehen.

3.2.3 Kardinalität der Tabellen

Der letzte Schritt von der Analyse des Statistiktool-Schemas ist die Überprüfung von der Größe einzelner Tabellen.

Die Tabellen BENUTZER und MCT_TIME sind die einzigen Tabellen, die keine relevante Information für das Statistiktool enthalten und deshalb nicht berücksichtigt werden müssen.

Die Tabellen MCT_UNI (Kardinalität: 10), MCT_KATEGORIEN (Kardinalität: 123), NATION (Kardinalität: 2) und TOOLS (Kardinalität: 15) gehören zu den Tabellen, deren Kardinalität gering und konstant bleibt. Die Kardinalitäten dieser Tabellen verändern sich nur, falls eine neue Universität bzw. Fachhochschule, eine neue Kategorie im MCT, eine neue Sprache oder ein neuer Trainer zu dem EDB-Portal hinzugefügt wird.

Die Tabellen MCT_FEEDBACK (Kardinalität: 279), SQL_OPTIMIZER_SCHEMA (Kardinalität: 131) und XQUERY_SCHEMA (Kardinalität: 98) haben auch eine geringe Kardinali-

tät, aber sie gehören zu den Tabellen, in die Datensätze immer wieder eingefügt werden können. Die geringe Kardinalität dieser Tabellen liegt in der geringen Verwendung des SQL-Abfrageoptimierungstools und des XQuery-Trainers begründet und dass die Benutzer nicht viele Feedbacks im MCT hinterlassen.

Die Tabellen MCT_HILFE (Kardinalität: 9.347), MCT_UNI_AUSGEWAEHLT (Kardinalität: 6.442), SQL_TRAINER_SCHEMA (Kardinalität: 12.784), SQL_TRAINER_1_SCHEMA (Kardinalität: 15.555) und TOOL_BEENDET (Kardinalität: 13.594) haben dagegen eine mittlere Kardinalität. Daraus kann man erkennen, dass der SQL-Trainer und der SQL-Trainer2 nicht so häufig von Benutzern verwendet werden. In die Tabelle MCT_HILFE werden Daten gespeichert, immer wenn die Erklärung bzw. die Hilfe im MCT aufgerufen wird. Die Tabelle MCT_UNI_AUSGEWAEHLT enthält nur zusätzliche Informationen über die ausgewählten Hochschulen im MCT.

Eine hohe Kardinalität haben die Tabellen MCT_KATEGORIEN_AUSGEWAEHLT (Kardinalität: 44.289), NATION_AUFGERUFEN (Kardinalität: 55.210), TOOL_AUFGERUFEN (Kardinalität: 67.106) und FRAGENAUSWERTUNG (Kardinalität: 262.847), weil in diesen Tabellen Daten gespeichert werden, die häufig bei der Verwendung des EDB-Portals gesammelt werden. Die größte Tabelle des Statistiktool-Schemas ist die Tabelle FRAGENAUSWERTUNG. In diese Tabelle werden die Ergebnisse für den MCT, SQL-Trainer, SQL-Trainer2 und XQuery-Trainer eingetragen. In die Tabellen MCT_KATEGORIEN_AUSGEWAEHLT und NATION_AUFGERUFEN werden Daten gespeichert, wenn ein Benutzer eine Kategorie im MCT bzw. eine Sprache im EDB-Portal auswählt. In die Tabelle TOOL_AUFGERUFEN wird immer, wenn ein Trainer oder Tool auf dem EDB-Portal gestartet wird, ein neuer Datensatz hinzugefügt.

	TABLE_NAME	NUM_ROWS
1	BENUTZER	1
2	FRAGENAUSWERTUNG	262847
3	MCT_FEEDBACK	279
4	MCT_HILFE	9347
5	MCT_KATEGORIEN	123
6	MCT_KATEGORIEN_AUSGEWAEHLT	44289
7	MCT_TIME	2250
8	MCT_UNI	10
9	MCT_UNI_AUSGEWAEHLT	6442
10	NATION	2
11	NATION_AUFGERUFEN	55210
12	SQL_OPTIMIZER_SCHEMA	131
13	SQL_TRAINER_SCHEMA	12784
14	SQL_TRAINER_1_SCHEMA	15555
15	TOOL_AUFGERUFEN	67106
16	TOOL_BEENDET	13594
17	TOOLS	15
18	XQUERY_SCHEMA	98

Abbildung 57: Tabellen des Statistiktool-Schemas mit ihren entsprechenden Kardinalitäten

(Stand: 06.04.2013, 17:30 Uhr)

3.3 Erkenntnisse aus der Analyse des Statistiktool-Schemas

In diesem Kapitel werden die gewonnenen Erkenntnisse aus der Analyse des Statistiktool-Schemas zusammengefasst. Die wichtigen Erkenntnisse liegen im Bereich der Indizierung, des Inhalts und der Kardinalität der Tabellen.

Die Verwendung von Indizes in den Tabellen und das Wachstum der Tabellen haben eine wichtige Bedeutung für das Statistiktool im Zusammenhang mit der Performance, weil diese beiden Eigenschaften einen großen Einfluss auf einen schnellen Zugriff der Daten haben. Im Kapitel „3.3.4.1 Annahme eines Performance-Problems“ wird ein mögliches Performance-Problem im Statistiktool thematisiert. Eine Lösung für dieses Performance-Problem wird im Kapitel „3.3.4.2 Lösungsansatz zum Performance-Problem“ präsentiert.

Ein wichtiger Teil der Analyse war es aber, herauszufinden, welche Daten in den Tabellen gespeichert sind, aus denen die Statistiken erstellt werden können. Dafür wurden diese Daten zuerst ausgewertet und dann nach Merkmalen gruppiert.

Dazu konnten auch einige Unstimmigkeiten im Statistiktool-Schema erkannt werden, die im Kapitel „3.3.5 Anmerkungen über das Statistiktool-Schema“ beschrieben werden.

3.3.1 Erkenntnisse zu Indizes in den Tabellen

Wie im Kapitel „3.2.2 Indizes und Primärschlüssel der Tabellen“ schon erläutert wurde, enthalten die meisten Tabellen des Statistiktool-Schemas mit den gesammelten Daten keinen Primärschlüssel oder Index. Der Grund dafür ist, dass diese Tabellen nur für das Sammeln von Daten erstellt wurden. Das bedeutet, dass die Daten in die Tabellen nur geschrieben und nicht gelesen werden mussten. Das Schreiben von Daten erfolgt in Tabellen ohne Indizes schneller als in Tabellen mit Indizes. Deswegen gab es für die Studenten-Gruppe, die für das Statistiktool-Schema zuständig war, auch keinen Grund, Indizes zu erstellen.

Für das Statistiktool können durch das Fehlen von Indizes in den Tabellen Nachteile entstehen, weil Performance-Probleme auftreten können. Das Problem liegt darin, dass das Lesen von Daten aus nicht-indizierten Spalten langsamer durchgeführt wird. Im Statistiktool müssen die Daten aus dem Statistiktool-Schema gelesen werden, um daraus die Statistik-Diagramme zu erzeugen. Das Verändern oder Einfügen von Daten ist im Statistiktool nicht möglich. Wenn die Erstellung und die Darstellung der Diagramme langsam ausgeführt werden, werden die Benutzer des Statistiktools dies als benutzerunfreundlich empfinden.

Die einfache Erstellung von Indizes in die Tabelle des Statistiktool-Schemas kann aber auch eine verlangsamte Durchführung des Datensammelns verursachen, weil das Sammeln von Daten viele INSERT-Anweisungen erfordert und dadurch viele Daten in die Tabellen geschrieben werden. Eine Verlangsamung des Datensammelns kann dazu führen, dass der Start bzw. die Verwendung der Trainer auf dem EDB-Portal langsam wird. In dieser Situation wird die Verwendung der Trainer durch die Benutzer beeinträchtigt und die Trainer werden als benutzerunfreundlich gelten.

3.3.2 Erkenntnisse zum Wachstum der Tabellenkardinalität

Aus der Analyse des Statistiktool-Schemas konnten auch wichtige Erkenntnisse über die Größe und das Wachstum der Tabellen gewonnen werden. Die Kardinalität bei einigen Tabellen wächst mit der Zeit sehr stark. Erst seit dem 02.04.2012 werden Daten von Trainern des EDB-Portals im Statistiktool-Schema gesammelt und nach einem Jahr enthalten einige Tabellen schon sehr viele Datensätze.

Wie im Kapitel „3.2.3 Kardinalität der Tabellen“ schon beschrieben wurde und in der Abbildung 57 „Tabellen des Statistiktool-Schemas mit ihren entsprechenden Kardinalitäten“ gezeigt wird, sind die Tabellen MCT_KATEGORIEN_AUSGEWAEHLT, NATION_AUFGERUFEN, TOOL_AUFGERUFEN und FRAGENAUSWERTUNG die größten Tabellen im Statistiktool-Schema. Diese Tabellen enthalten auch sehr wichtige Informationen für das Statistiktool.

Aus diesem Grund mussten die Daten als Diagramme im Statistiktool angezeigt werden, damit die Informationen aus den Daten schneller erfasst und auch besser verglichen werden können. Eine so große Menge an Daten in einer tabellarischen Form darzustellen, wäre unübersichtlich und deshalb nicht sinnvoll.

3.3.3 Erkenntnisse zum Tabelleninhalt

Die Erkenntnisse aus der Analyse des Tabelleninhalts kamen zuerst mit der Überprüfung, welche Informationen die Datensätze liefern und zu welchem Trainer bzw. Tool des EDB-Portals sie gehören. Daraus konnte eine Informationsstruktur für das Statistiktool gebildet werden.

Die Analyse der Datensätze ergab auch, dass nicht alle für das Statistiktool notwendigen Daten im Statistiktool-Schema gespeichert sind. Somit mussten einige Daten aus anderen Schemata entnommen werden, um die Darstellung der Daten im Statistiktool zu vervollständigen.

Durch die Analyse der Datensätze erkannte man auch, dass die Anforderung an das Statistiktool, die darzustellenden Daten nach Hochschule zu trennen, nur beim MCT erfüllt werden konnte.

In den folgenden Kapiteln werden die einzelnen Erkenntnisse zum Tabelleninhalt erläutert.

3.3.3.1 Informationsstruktur im Statistiktool

Die gesammelten Daten konnten nach Informationen zu einem bestimmten Trainer getrennt und in Gruppen eingeteilt werden, aus denen insgesamt sechs Datengruppen entstanden. Eine Gruppe enthält allgemeine Informationen über alle Trainer, während die anderen Gruppen trainerspezifisch sind. Zu den Trainern zählen der MCT, SQL-Trainer, SQL-Trainer2, XQuery-Trainer und das SQL-Abfrageoptimierungstool.

Für jede Gruppe wurden Fragen gebildet, die mit den Informationen aus den Datensätzen beantwortet werden können. Ein Beispiel für eine Frage lautet „Welches Tool wird am meisten

aufgerufen?“, die zur Gruppe mit Informationen über alle Trainer gehört. Diese Fragen sollen für die Benutzer als Option in der Einstellung bzw. Konfiguration der Datenabfragen im Statistiktool übernommen werden. Aus diesem Grund wurden sie verkürzt und aussagekräftiger gemacht. So können ihre Bedeutungen von den Benutzern schneller verstanden werden. Die Frage des oberen Beispiels „Welches Tool wird am meisten aufgerufen?“ wurde dann in den Satz „Tools mit der größten Anzahl von Aufrufen“ verändert.

Mit der Aufteilung der gewonnenen Informationen aus den Daten in Gruppen konnte eine Struktur aufgebaut werden, die im Statistiktool eingesetzt wird. Die Gruppen bilden die Hauptbereiche der Informationsstruktur. Zu jeder Gruppe gehören die verschiedenen Optionen zur Datenabfrage. Für die Benutzer bedeutet es, dass sie im Statistiktool zuerst eine Gruppe auswählen müssen. Danach stehen dann alle Optionen der ausgewählten Gruppe zur Verfügung, damit sie die Daten abfragen können.

Im Folgenden werden alle sechs Gruppen mit den entsprechenden Optionen zur Datenabfrage im Statistiktool aufgelistet. Dazu werden auch die Tabellen mit den Spalten gezeigt, die die benötigten Informationen enthalten.

Gruppe: Alle Tools

Diese Gruppe enthält Informationen über alle Tools bzw. Trainer. Diese Informationen wurden aus den Datensätzen folgender Tabellen entnommen:

- TOOLS (ID, NAME)
- TOOL_AUFGERUFEN (TOOL_ID, DATUM)
- TOOL_BEENDET (TOOL_ID, DATUM)
- NATION (ID, NATIONNAME)
- NATION_AUFGERUFEN (NATION_ID, DATUM)

Aus den Informationen konnten diese Optionen erstellt werden:

- Tools mit der kleinsten Anzahl von Aufrufen
- Tools mit der größten Anzahl von Aufrufen
- Tools, die am wenigsten beendet wurden
- Tools, die am meisten beendet wurden
- Ausgewählte Sprachen

Gruppe: MCT

Diese Gruppe enthält Informationen über den MCT. Diese Informationen wurden aus den Datensätzen folgender Tabellen entnommen:

- MCT_UNI (ID, UNI_NAME)
- MCT_UNI_AUSGEWAEHLT (MCT_UNI_ID, SESSION_ID, DATUM)
- MCT_FEEDBACK (FRAGE_ID, DATUM)
- MCT_HILFE (FRAGE_ID, DATUM)
- MCT_KATEGORIEN (ID, UNI_ID, KATNR, BEZEICHNUNG)
- MCT_KATEGORIEN_AUSGEWAEHLT (MCT_KAT_ID, DATUM)
- FRAGENAUSWERTUNG (ID, TOOL_ID, FRAGE_ID, RICH-TIG_BEANTWORTET, ZEIT, DATUM)

Aus den Informationen konnten diese Optionen erstellt werden:

- Ausgewählte Hochschulen
- Aufgaben mit der größten Anzahl von richtigen Antworten
- Aufgaben mit der größten Anzahl von falschen Antworten
- Aufgaben mit der kürzesten Lösungszeit
- Aufgaben mit der längsten Lösungszeit
- Aufgaben mit der kleinsten Anzahl von Erklärungsaufrufen
- Aufgaben mit der größten Anzahl von Erklärungsaufrufen
- Aufgaben mit der kleinsten Anzahl von Feedbackaufrufen
- Aufgaben mit der größten Anzahl von Feedbackaufrufen
- Aufgaben-Kategorien, die am wenigsten ausgewählt werden
- Aufgaben-Kategorien, die am meisten ausgewählt werden

Gruppe: SQL-Trainer

Diese Gruppe enthält Informationen über den SQL-Trainer. Diese Informationen wurden aus den Datensätzen folgender Tabellen entnommen:

- SQL_TRAINER_1_SCHEMA (AUFGABE_ID, SCHEMA_ID, DATUM)
- FRAGENAUSWERTUNG (ID, TOOL_ID, FRAGE_ID, RICH-TIG_BEANTWORTET, ZEIT, DATUM)

Aus den Informationen konnten diese Optionen erstellt werden:

- Schema mit der kleinsten Anzahl von Aufrufen
- Schema mit der größten Anzahl von Aufrufen
- Aufgaben mit der größten Anzahl von richtigen Antworten
- Aufgaben mit der größten Anzahl von falschen Antworten

Gruppe: SQL-Trainer2

Diese Gruppe enthält Informationen über den SQL-Trainer. Diese Informationen wurden aus den Datensätzen folgender Tabellen entnommen:

- SQL_TRAINER_SCHEMA (AUFGABE_ID, SCHEMA_ID, DATUM)
- FRAGENAUSWERTUNG (ID, TOOL_ID, FRAGE_ID, RICHTIG_BEANTWORTET, ZEIT, DATUM)

Aus den Informationen konnten diese Optionen erstellt werden:

- Schema mit der kleinsten Anzahl von Aufrufen
- Schema mit der größten Anzahl von Aufrufen
- Aufgaben mit der größten Anzahl von richtigen Antworten
- Aufgaben mit der größten Anzahl von falschen Antworten

Gruppe: XQuery-Trainer

Diese Gruppe enthält Informationen über den SQL-Trainer. Diese Informationen wurden aus den Datensätzen folgender Tabellen entnommen:

- XQUERY_SCHEMA (AUFGABE_ID, SCHEMA_ID, DATUM)
- FRAGENAUSWERTUNG (ID, TOOL_ID, FRAGE_ID, RICHTIG_BEANTWORTET, ZEIT, DATUM)

Aus den Informationen konnten diese Optionen erstellt werden:

- Schema mit der kleinsten Anzahl von Aufrufen
- Schema mit der größten Anzahl von Aufrufen
- Aufgaben mit der größten Anzahl von richtigen Antworten
- Aufgaben mit der größten Anzahl von falschen Antworten

Gruppe: SQL-Abfrageoptimierungstool

Diese Gruppe enthält Informationen über den SQL-Trainer. Diese Informationen wurden aus den Datensätzen folgender Tabellen entnommen:

- SQL_OPTIMIZER_SCHEMA (AUFGABE_ID, SCHEMA_ID, DATUM)

Aus den Informationen konnten diese Optionen erstellt werden:

- Schema mit der kleinsten Anzahl von Aufrufen
- Schema mit der größten Anzahl von Aufrufen
- Schema mit den meisten ausgewählten SELECT-Abfragen
- Schema mit den meisten geschriebenen SELECT-Abfragen
- Aufgaben mit der kleinsten Anzahl von Aufrufen
- Aufgaben mit der größten Anzahl von Aufrufen

3.3.3.2 Datenergänzung für das Statistiktool

Eine andere wichtige Erkenntnis aus der Analyse der Daten war es auch, dass im Statistiktool-Schema nicht alle Daten gespeichert sind, die das Statistiktool benötigt. Für einige Daten des Statistiktool-Schemas müssen zusätzliche Daten aus anderen Schemata entnommen werden, um die Daten im Statistiktool zu ergänzen. Viele Tabellen des Statistiktool-Schemas enthalten nur die ID der Schemata und die ID der Aufgaben. Die Namen der Schemata und die Aufgabenstellungen selbst wurden aber nicht mitgespeichert. Die einzige Ausnahme ist die Tabelle SQL_TRAINER_1_SCHEMA, in der die ID der Schemata bereits enthalten sind. Diese zusätzlichen Daten werden aus den Schemata entnommen, die in der folgenden Tabelle vorgeführt werden.

Statistiktool-Schema	Andere Schemata		
Spalte (Tabelle)	**Schema**	**Tabelle**	**Spalte**
FRAGE_ID (STATS_MCT)	PLISCHKE_PROJEKT	FRAGEN	TEXT
MCT_UNI_ID (STATS_MCT)	PLISCHKE_PROJEKT	BEREICH	NAME
FRAGE_ID (SQL_OPTIMIZER_ SCHEMA)	SQLOPTIMIZER	SQLABFRAGEN	SQLABFRAGE
SCHEMA_ID (SQL_OPTIMIZER_SCHEMA)	SQLOPTIMIZER	DB	NAME
FRAGE_ID (SQL_TRAINER_ SCHEMA)	PHILIPP	AUFGABEN	AUFGABENTEXT
SCHEMA_ID (SQL_TRAINER_SCHEMA)	PHILIPP	DBSCHEMA	NAME
FRAGE_ID (SQL_TRAINER_1_SCHEMA)	SQLTRAINER	AUFGABE	AUFGABE
FRAGE_ID (XQUERY_ SCHEMA)	PHILIPP_PROJEKT	AUFGABEN	AUFGABENTEXT
SCHEMA_ID (XQUERY_SCHEMA)	PHILIPP_PROJEKT	XMLFILES	NAME

Tabelle 6: Schemata, aus denen die zusätzlichen Daten entnommen werden

3.3.3.3 Anforderung zur Datentrennung nach Hochschule

Die Analyse des Tabelleninhalts hat auch gezeigt, dass eine Trennung der Daten nach Hochschulen nur beim MCT möglich ist. Die Fragen vom MCT werden in die Tabelle FRAGEN aus dem Datenbankschema PLISCHKE_PROJEKT gespeichert und nach Bereichen aufgeteilt. Jeder Bereich entspricht einer Hochschule, die ihre eigenen Fragen hat. In diesem Schema sind die Bereiche und Hochschulen in die Spalten BEREICH_ID bzw. NAME aus der Tabelle BEREICHE gespeichert. Die Tabelle FRAGEN enthält unter anderem die Spalten FRAGE_ID und BEREICH_ID.

Durch eine Tabellenverbindung zwischen den Tabellen BEREICHE und FRAGE können dann die Daten von einzelnen Hochschulen eindeutig identifiziert werden.

Bei den anderen Trainern ist eine Trennung der Daten nach Hochschulen nicht möglich, weil die Fragen nicht nach Bereich bzw. Hochschule getrennt werden und bei den Zugangsdaten der Benutzer nicht gespeichert wird, wo die Studenten studieren. Das EDB-Portal ist auch ein offener Dienst, auf dem sich jeder registrieren kann, auch wenn man nicht zu einer Hochschule gehört. So kann das Sammeln von Daten auch nicht nach Hochschulen sortiert werden.

3.3.4 Performance-Problem im Statistiktool

3.3.4.1 Annahme eines Performance-Problems

Mit den Erkenntnissen aus den Kapiteln „3.3.1 Erkenntnisse zu Indizes in den Tabellen“ und „3.3.2 Erkenntnisse zum Wachstum der Tabellenkardinalität“ steht fest, dass die wichtigsten Tabellen des Statistiktool-Schemas für das Statistiktool keinen Index besitzen und dass sie in kurzer Zeit sehr groß werden.

Als Beispiel für eine große Tabelle wird die Tabelle FRAGENAUSWERTUNG betrachtet. Sie enthält nach einem Jahr ungefähr 260.000 Zeilen. In fünf Jahren kann man davon ausgehen, dass diese Tabelle 1.300.000 Datensätze und in den nächsten zehn Jahren sogar 2.600.000 Datensätze enthalten könnte.

Durch diese Schätzung und das Fehlen von Indizes kommt man zu der Erkenntnis, dass die Ausführung von SELECT-Anweisungen im Statistiktool mit so vielen Datensätzen sehr lang dauern kann und den Datenbank-Server sehr überlasten kann. Das führt zu einer Verlangsamung des Statistiktools und infolgedessen zu Performance-Problemen.

Zur Überprüfung eines möglichen Performance-Problems im Statistiktool wurde ein Ausführungsplan von einer SELECT-Anweisung analysiert, die auf die gesammelten Daten zugreift.

Im Programm Oracle SQL Developer steht ein Feature zur Verfügung, das den Ausführungsplan einer SELECT-Anweisung anzeigt.
Ein Ausführungsplan (Query Execution Plan) wird vom Oracle-Optimizer erzeugt und stellt eine detaillierte Zugriffsreihenfolge auf Daten, Indizes, Tabellen und andere Objekte einer Datenbank dar. Diese Zugriffsreihenfolge zeigt, wie die Ausführung einer SELECT-Anweisung erfolgt und wird durch verschiedene Ausführungsplanoperationen beschrieben. Die wichtigsten Ausführungsplan-Operationen sind die Zugriffs-, Join-, Sortier- und Mengenoperationen.

Der Oracle-Optimizer sucht nach einer effizienten Methode, um eine SELECT-Anweisung auszuführen. Der Oracle-Optimizer versucht die Ausführung zu optimieren, indem er unterschiedliche Ausführungspläne für die SELECT-Anweisung berechnet und den kostengünstigsten Ausführungsplan auswählt. Die Berechnung der Ausführungspläne erfolgt durch die Schätzung von Kosten (Verbrauch von Ressourcen) der einzelnen Ausführungsplanoperationen. Dafür benötigt der Oracle-Optimizer Statistiken, die die Datenverteilung und Speichereigenschaften von Tabellen, Spalten, Indizes, Partitionen und anderen Objekten der Datenbank sowie Systemdaten, wie I/O-Zugriff (Input/Output) und die CPU-Nutzung erfassen. Solche Informationen werden gesammelt und im DATA DICTIONARY gespeichert.

Der ausgewählte Ausführungsplan wird dann zur Ausführung der SELECT-Anweisung verwendet. Ein Ausführungsplan gilt als kostengünstigster, wenn die Ergebnismenge der SELECT-Anweisung in kürzester Zeit und mit der geringsten Nutzung von Ressourcen zurückgegeben wird.

Auf der folgenden Abbildung wird eine SELECT-Anweisung als Beispiel gezeigt, die die Anzahl von Aufrufen und von Beenden aller Trainer aus den Tabellen TOOL_AUFGERUFEN und TOOL_BEENDEN zwischen 02.04.2012 und 30.06.2013 berechnet. Die SELECT-Anweisung berechnet die Anzahl von Aufrufen mit der Funktion COUNT und enthält eine verschachtelte SELECT-Anweisung, die das Beenden der Trainer berechnet.

```
SELECT TO_CHAR(ta.datum, 'DD.MM.YYYY'),
       ta.tool_id,
       COUNT(ta.tool_id) AS aufgerufen,
       (SELECT COUNT(tb.tool_id)
        FROM tool_beendet tb
        WHERE TO_CHAR(ta.datum, 'DD.MM.YYYY')=TO_CHAR(tb.datum, 'DD.MM.YYYY')
        AND ta.tool_id=tb.tool_id) AS beendet
FROM tool_aufgerufen ta WHERE ta.datum BETWEEN '02.04.2012' AND '30.06.2013'
GROUP BY TO_CHAR(ta.datum, 'DD.MM.YYYY'), ta.tool_id;
```

Abbildung 58: SELECT-Anweisung zum Laden der gesammelten Daten

Auf der folgenden Abbildung wird nun der Ausführungsplan der SELECT-Anweisung mit dem Programm Oracle SQL Developer gezeigt, der in die globale temporäre Tabelle PLAN_TABLE gespeichert worden ist.

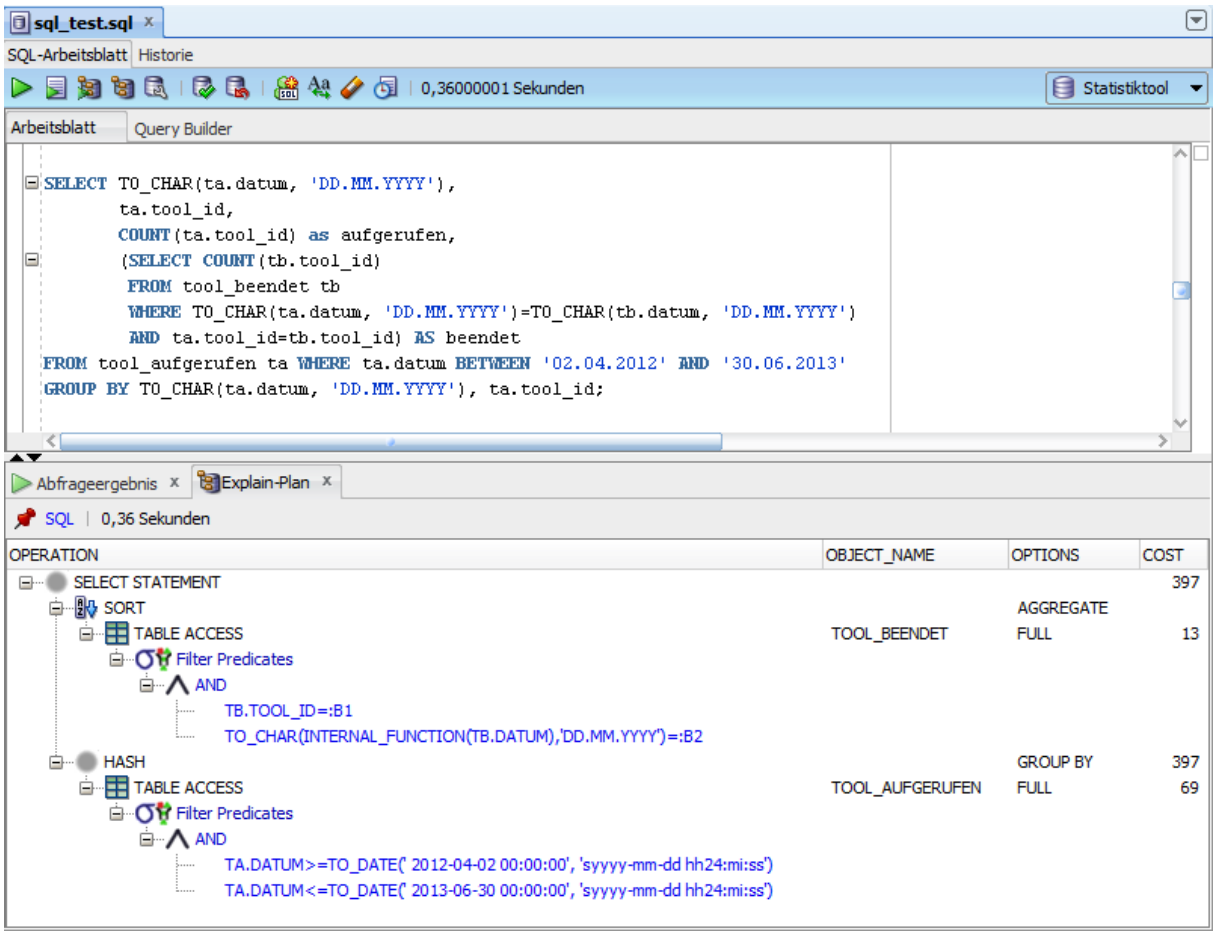

Abbildung 59: Ausführungsplan der SELECT-Anweisung im Oracle SQL Developer

Aus der Abbildung 59 kann man die gesamten Kosten für die Ausführung der SELECT-Anweisung entnehmen, deren Ausführungsplan vom Oracle-Optimizer ausgewählt wurde. Diese Kosten wurden auf 397 geschätzt und stehen in der Spalte COST. Der Wert der Kosten hat keine besondere Maßeinheit. Es ist nur ein gewichteter Wert, damit die Kosten zwischen den Ausführungsplänen verglichen werden können.

Zur Tabelle PLAN_TABLE mit dem Ausführungsplan gehören auch die Spalten OPERATION, OBJECT_NAME und OPTIONS. In der Spalte OBJECT_NAME stehen die Namen der Objekte, auf das sich die Operationen beziehen. Ein Objekt ist normalerweise eine Tabelle, ein Index oder auch eine View. In diesem Ausführungsplan wird auf die Tabellen TOOL_AUFRUFEN und TOOL_BEENDEN zugegriffen.

Die Spalten OPERATION und OPTIONS zeigen im Ausführungsplan der SELECT-Anweisung, welche Ausführungsplanoperationen durchgeführt wurden. Dieser Ausführungsplan besteht aus den Operationen SELECT STATEMENT, SORT AGGREGATE, TABLE ACCESS FULL und HASH GROUP BY.

Die Ausführungsplanoperation SELECT STATEMENT zeigt nur, dass der Ausführungsplan aus einer SELECT-Anweisung stammt. Die Operation SORT AGGREGATE weist auf eine Aggregationsfunktion, wie z. B. COUNT(), in der SELECT-Anweisung hin. Die Operation HASH GROUP BY deutet auf eine Gruppierung mit der Verwendung von einer Hash-Tabelle in der SELECT-Anweisung an.

Die wichtige Operation, die im Ausführungsplan beachtet werden muss, ist die Tabellenzugriffsoperationen TABLE ACCESS FULL. Durch diese Tabellenzugriffsoperationen ist die Performance nicht optimal, weil diese Operation alle Datensätze von einer Tabelle liest und sie gegen die WHERE-Klausel prüft.

Die Operation TABLE ACCESS FULL gehört zu den systembelastenden Operationen und liefert eine sehr schlechte Performance bei großen Tabellen. Der Oracle-Optimizer musste aber diese Operation verwenden, weil die Tabellen keinen Index haben, und damit die Verwendung einer Indexzugriffsoperation, wie z. B. INDEX RANGE SCAN, nicht möglich war. Ein Indexzugriff ist deutlich kostengünstiger als ein Tabellenzugriff. Unnötige TABLE AC-

CESS FULL-Operationen können aus dem Ausführungsplan entfernt werden, indem Indizes in die Tabellen gesetzt werden.

Die Verwendung von der Operation TABLE ACCESS FULL ist nur bei kleinen Tabellen sinnvoll oder, wenn die SELECT-Anweisung einen großen Teil der Tabelle als Ergebnismenge zurückgibt. Beide Fälle treffen aber auf die Tabellen des Statistiktool-Schemas und die SELECT-Anweisungen im Statistiktool nicht zu, weil die meisten Tabellen viele Datensätze enthalten und ein Zugriff auf einen großen Teil der Tabelle nicht erfolgt. Die SELECT-Anweisungen im Statistiktool greifen immer auf Daten aus nur einem bestimmten Zeitraum und nicht auf alle Daten einer Tabelle zu.

Wie im Kapitel „3.3.1 Erkenntnisse zu Indizes in den Tabellen“ schon erläutert wurde, können Indizes in die Tabellen des Statistiktool-Schemas nicht einfach erstellt werden, weil sonst das Sammeln der Daten und dadurch auch die Verwendung von den Trainern auf dem EDB-Portal verlangsamt werden können.

Im nächsten Kapitel wird ein Lösungsansatz für dieses Performance-Problem präsentiert.

3.3.4.2 Lösungsansatz zum Performance-Problem

Für das im vorherigen Kapitel beschriebene Performance-Problem musste ein Lösungsansatz entwickelt werden, der dem Statistiktool einen schnellen Zugriff auf die Daten ermöglicht, aber auch gleichzeitig keine Verlangsamung des Datensammelns auf dem EDB-Portal verursacht.

Die im Statistiktool eingesetzte Lösung sieht vor, dass das Sammeln von Daten und die Tabellen im Statistiktool-Schema nicht verändert werden. So können die gesammelten Daten weiterhin auf Tabellen gespeichert werden, die keine Indizes enthalten.

Zur Lösung gehört auch, dass die gesammelten Daten einmalig berechnet werden, indem sie nach Datum und noch einem bestimmten Merkmal (z. B.: TOOL_ID oder SCHEMA_ID) gruppiert werden. Die berechneten Daten werden dann in neu angelegten Tabellen gespei-

chert. Die neu angelegten Tabellen werden Statistik-Tabellen genannt und werden zu den übrigen Tabellen vom Statistiktool-Schema hinzugefügt. Sie enthalten die berechneten Daten, die als Statistik-Daten bezeichnet werden. Auf diese Statistik-Daten greift nun das Statistiktool zu, um die Diagramme zu erstellen. Die Statistik-Tabellen und die Statistik-Daten werden im Kapitel „4.2.1 Beschreibungen der Statistik-Tabellen“ ausführlich beschrieben.

Der Vorteil dieser Lösung liegt darin, dass das Sammeln von Daten nicht beeinträchtigt wird, und dass das Statistiktool auf Tabellen zugreifen kann, die deutlich kleiner als die originalen Tabellen mit den gesammelten Daten sind. Außerdem enthalten die Statistik-Tabellen durch die Primärschlüssel Indizes, um einen schnellen Zugriff auf die Statistik-Daten zu gewährleisten.

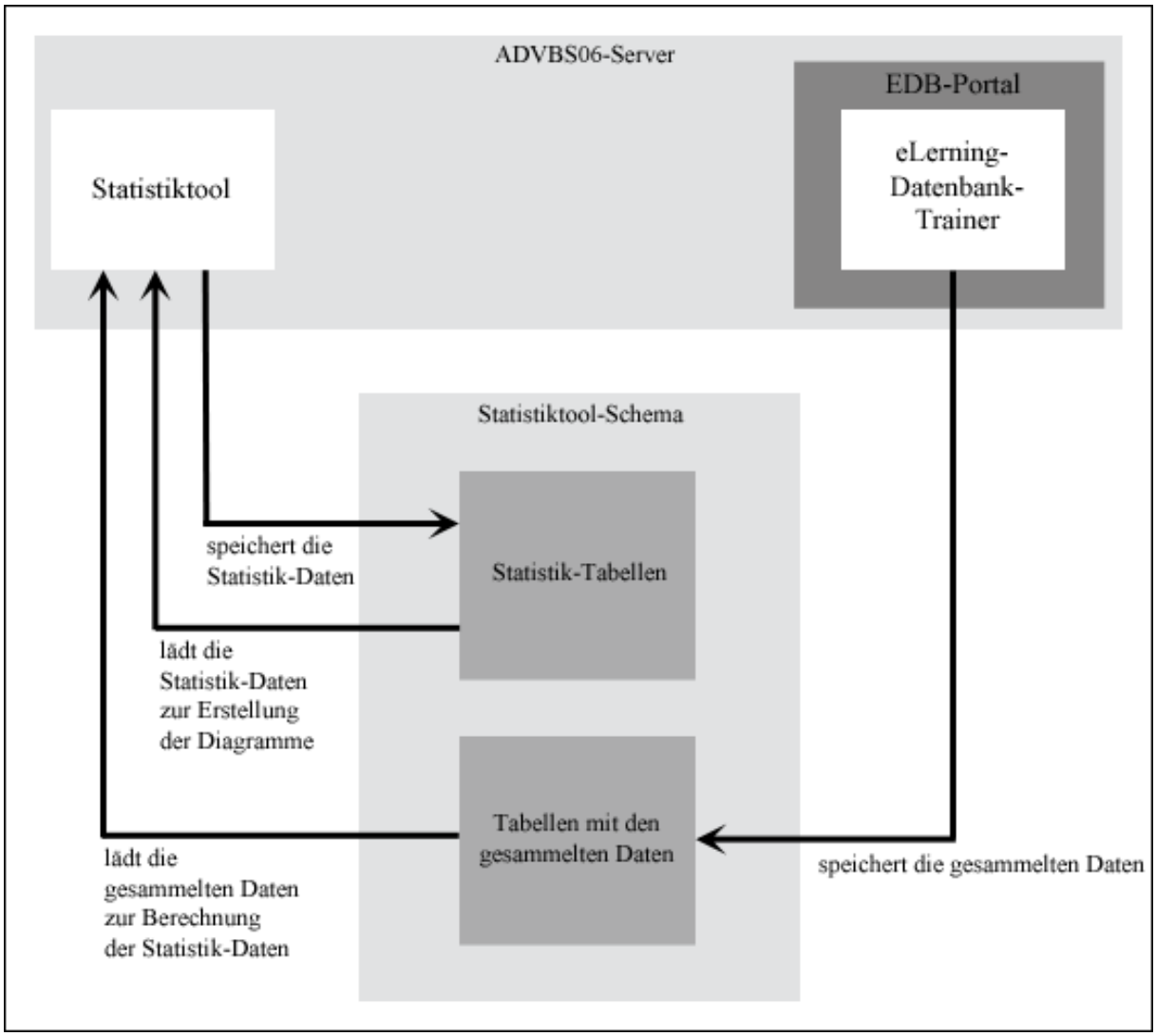

Abbildung 60: Zugriffübersicht auf das Statistiktool-Schema

Der verschaffte Zugriffsvorteil auf die Statistik-Tabelle kann in einem Beispiel verdeutlicht werden. Als Beispiel wird die Tabelle TOOL_AUFGERUFEN mit den gesammelten Daten genommen, für die die Statistik-Tabelle STATS_TOOL_AUFGERUFEN erstellt wurde. Die Tabelle TOOL_AUFGERUFEN ist die zweitgrößte Tabelle des Statistiktool-Schemas und enthält nach einem Jahr schon ungefähr 67.000 Zeilen. Die Datensätze dieser Tabelle sollen nach den Spalten DATUM und TOOL_ID gruppiert werden, um die Anzahl von Aufrufen eines Trainers innerhalb eines Tages zu berechnen.

Die Statistik-Tabelle STATS_TOOL_AUFGERUFEN enthält deutlich wenige Datensätze als die Tabelle TOOL_AUFGERUFEN. Der Worst Case (ungünstigster Fall) für die Statistik-Tabelle liegt in diesem Fall vor, wenn alle Trainer von dem EDB-Portal jeden Tag aufgerufen werden.

Worst Case: 365 Tage x 16 Tools = 5.840 erzeugte Zeilen pro Jahr.

Prozentsatz von erzeugten Zeilen pro Jahr: (5.840 * 100) / 67.000 = 8,71%

Abbildung 61: Geschätzte Anzahl von jährlich erzeugten Zeilen in Statistik-Tabellen

In Prozentsatz entsprechen die 5.840 erzeugten Zeilen (Statistik-Daten) nur 8,71% von den aktuellen 67.000 Zeilen (gesammelte Daten) der Tabelle TOOL_AUFGERUFEN. Selbst beim Worst Case würde das Statistiktool auf die Statistik-Tabellen zugreifen, die immer weniger als 10% der Datensätze von den originalen Tabellen enthalten.

Wenn man die Mengen der Datensätze betrachtet, die nach mehreren Jahren gesammelt werden, ist es vorteilhaft, auf kleinere Datenmengen zuzugreifen, um den Datenbank-Server nicht zu überlasten und die Ausführung des Statistiktools zu beschleunigen.

Die folgende Abbildung zeigt eine SELECT-Anweisung, die der SELECT-Anweisung aus dem Beispiel des Kapitels „3.3.4.1 Annahme eines Performance-Problems“ entspricht. Diese SELECT-Anweisung ist nicht so komplex aufgebaut und greift nicht auf die Tabellen TOOL_AUFGERUFEN und TOOL_BEENDET mit den gesammelten Daten, sondern auf die

entsprechende Statistik-Tabelle STATS_TOOL_AUFGERUFEN mit den Statistik-Daten zu. Beide SELECT-Anweisungen liefern die gleiche Ergebnismenge.

```
SELECT *
FROM STATS_TOOL_AUFGERUFEN
WHERE datum BETWEEN '02.04.2012' AND '30.06.2013';
```

Abbildung 62: SELECT-Anweisung, die auf die Statistik-Tabelle zugreift

Die Performance von dieser SELECT-Anweisung wird durch die Analyse des Ausführungsplans überprüft. Die folgende Abbildung zeigt den Ausführungsplan von der SELECT-Anweisung, Zur Anzeige des Ausführungsplans wurde das Programm Oracle SQL Developer nochmal verwendet.

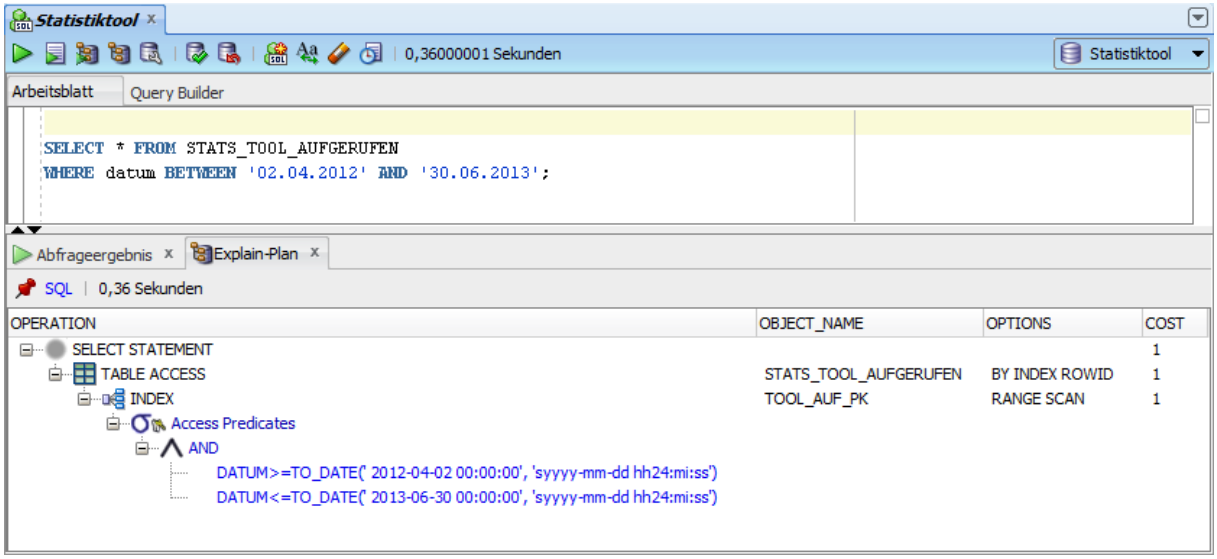

Abbildung 63: Ausführungsplan der SELECT-Anweisung im Oracle SQL Developer

Die Tabelle PLAN_TABLE mit dem Ausführungsplan zeigt in der Spalte COST, dass die Kosten für die Ausführung dieser SELECT-Anweisung deutlich kleiner sind. Sie wurden auf nur 1 geschätzt. Bei einem direkten Zugriff auf die Tabellen mit den gesammelten Daten betragen die Kosten 397.

Bei Betrachtung des Ausführungsplans erkennt man auch, dass er wesentlich weniger Ausführungsoperationen enthält. Dieser Ausführungsplan besteht aus den Operationen SELECT

STATEMENT, TABLE ACCESS BY INDEX ROWID und INDEX RANGE SCAN. Die Operation INDEX RANGE SCAN ist eine Indexzugriffsmethode, die die physikalischen Adressen (ROWID) der in der WHERE-Klausel ausgewählten Datensätze aus dem Index TOOL_AUF_PK ermittelt. Mit der Tabellenzugriffsmethode TABLE ACCESS BY INDEX ROWID wird dann die Ergebnismenge anhand der ROWIDs aus der Tabelle STATS_TOOL_AUFGERUFEN geladen. Die Ausführungsplanoperation SELECT STATEMENT zeigt nur, dass es sich beim Ausführungsplan um eine SELECT-Anweisung handelt.

Der Vergleich zwischen den Ausführungsplänen zeigt, dass die SELECT-Anweisung, die auf die Statistik-Tabellen zugreift, kostengünstiger ist, und dadurch die Ergebnismenge schneller zurückgeben kann. Der große Vorteil von diesem Ausführungsplan liegt darin, dass die systembelastende Operation TABLE ACCESS FULL nicht verwendet wird. Der Grund dafür ist, dass die zugegriffene Statistik-Tabelle einen Index besitzt. Wegen des Indexes kann der Oracle-Optimizer zuerst eine schnelle Indexzugriffsoperation durchführen, um die ROWIDs der ausgewählten Zeilen zu ermitteln. Danach werden die Daten mit den ROWIDs gezielt aus der Tabelle geladen. Bei der Operation TABLE ACCESS FULL muss die ganze Tabelle gelesen werden, was bei großen Tabellen zu Performance-Problemen führt.

Für das Berechnen und das Speichern von den Statistik-Daten musste eine neue Funktion im Statistiktool implementiert werden. Diese neue Funktion kann aber auch zur Überlastung des Datenbank-Servers führen, weil sie auf sehr große Tabellen mit den gesammelten Daten zugreifen muss, um die Statistik-Daten zu berechnen. Aus diesem Grund werden nur bei der ersten Berechnung alle gesammelten Daten einer Tabelle berücksichtigt. Alle Statistik-Daten werden dauerhaft in den Statistik-Tabellen gespeichert, damit diese Daten nur ein einziges Mal berechnet werden müssen. Wegen der ständigen Sammlung von neuen Daten ist aber eine regelmäßige Aktualisierung der Statistik-Tabellen notwendig. In diesem Fall werden nicht mehr alle gesammelten Daten von einer Tabelle in die Berechnung hineingezogen, sondern nur die neu gesammelten Daten, die nach der Berechnung auch in den Statistik-Tabellen abgelegt werden. Diese Daten befinden sich in einem bestimmten Zeitraum, der aus der Spalte DATUM abgelesen werden kann. Der Zeitraum liegt immer zwischen dem Datum der letzten Berechnung und dem Datum, in dem die neue Berechnung durchgeführt wird. Diese Informationen werden in die Tabelle STATS_AKT_STAND gespeichert.

Die Aktualisierung der Statistik-Tabellen soll mindestens einmal im Monat stattfinden, damit die Berechnung keinen so großen Umfang an gesammelte Daten berücksichtigen muss und der Datenbank-Server nicht überlastet wird.

Als einziger Nachteil soll erwähnt werden, dass das Berechnen und Speichern der Statistik-Daten noch mehr Daten erzeugen und dadurch auch mehr Speicherplatz benötigt wird.

3.3.5 Anmerkungen zum Statistiktool-Schema

3.3.5.1 Fehler in der Tabelle MCT_KATEGORIEN_AUSGEWAEHLT

Bei der Erstellung der Tabelle MCT_KATEGORIEN_AUSGEWAEHLT ist der Studenten-Gruppe ein Fehler unterlaufen. Mit diesem Fehler ist es nicht mehr möglich, die Bezeichnung einer Kategorie aus der Tabelle MCT_KATEGORIEN eindeutig zu identifizieren. Die Tabelle MCT_KATEGORIEN wurde von Datenbankadministratoren angelegt.

In die Tabelle MCT_KATEGORIEN_AUSGEWAEHLT (MCT_KAT_ID, DATUM) wird die UNI_ID aus der Tabelle MCT_KATEGORIEN (ID, UNI_ID, KATNR, BEZEICHNUNG) bei dem Sammeln von Daten nicht mitgespeichert. Somit erfolgt der benötigte Equi-Join zwischen den Tabellen nur durch die Spalten MCT_KAT_ID und KATNR, obwohl die Universitäten und Fachhochschulen ähnliche, aber nicht immer die gleichen Bezeichnungen für die gleichen Kategorien verwenden. Die UNI_ID aus der Tabelle MCT_KATEGORIEN entspricht der Spalte ID der Tabelle MCT_UNI (ID, UNI_NAME).

Zur Beseitigung dieses Fehlers war die einfachste Lösung, sich für nur eine Hochschule zu entscheiden und immer die ID dieser Hochschule in die WHERE-Klausel der SELECT-Anweisungen im Statistiktool zu verwenden. Die Entscheidung fiel auf die Fachhochschule Köln mit der ID 14.

3.3.5.2 Unterschiedliche Datentypen in den Spalten SCHEMA_ID

Im Statistiktool-Schema gibt es vier Tabellen, die eine Spalte SCHEMA_ID enthalten. Im Zusammenhang mit einer Datenbanktabelle ist es üblich, dass man eine Spalte SCHEMA_ID eigentlich mit dem Datentyp NUMBER einordnet. Aber in zwei Tabellen ist diese Spalte vom Datentyp VARCHAR2.

Im Kapitel „3.2.1 Inhalt und Beschreibung der Tabellen“ wurde schon dargestellt, dass die Spalte SCHEMA_ID von Datentyp NUMBER in den Tabellen SQL_TRAINER_SCHEMA und SQL_OPTIMIZER_SCHEMA ist. Dagegen ist die Spalte SCHEMA_ID in den Tabellen SQL_TRAINER_1_SCHEMA und XQUERY_SCHEMA vom Datentyp VARCHAR2 und diese müssen im Quellcode des Statistiktools gesondert behandelt werden.

Diese Unstimmigkeit bei den Tabellen SQL_TRAINER_1_SCHEMA und XQUERY_SCHEMA verlangt, dass bei der Berechnung der Statistik-Daten zuerst mit einer IF-Anweisung überprüft werden muss, um welche Tabelle es sich handelt. Bei der Tabelle XQUERY_SCHEMA wird die ID (Nummer) des Schemas als String gespeichert. Die Werte der Spalte können einfach mit der Funktion parseInt() in einen Integer konvertiert werden und in eine Spalte von Typ NUMBER in die entsprechende Statistik-Tabelle STATS_XQUERY_SCHEMA gespeichert werden.

Bei der Tabelle SQL_TRAINER_1_SCHEMA ist das nicht möglich, weil in die Spalte SCHEMA_ID nicht die ID des Schemas als Nummer, sondern der Name des Schemas (z. B. „Fahrrad“) selbst gespeichert wird. Somit konnten auch nur die Namen der Schemata in die Spalte SCHEMA_ID der Statistik-Tabelle STATS_SQL_TRAINER_1_SCHEMA bei der Berechnung der Statistik-Daten gespeichert werden.

3.4 Framework zur Erstellung von Diagrammen

Für das Statistiktool wurden zwei Frameworks analysiert, die die Eigenschaften besitzen, Diagramme in einer einfachen Form zu erstellen. Die Anwendung von APEX und Dojo werden untersucht, damit das vorteilhafte Framework ausgewählt wird. Dieses wird dann in das Statistiktool implementiert.

In diesem Kapitel wird zuerst APEX erläutert. Das Dojo Toolkit wurde bereits im Kapitel „2.2 DOJO Toolkit“ beschrieben. Im Kapitel „3.4.5 Vergleich zwischen APEX und DOJO“ werden der Vergleich und die Bewertung von APEX und Dojo zusammengefasst, aus denen die Auswahl für das Dojo Toolkit begründet wird.

3.4.1 APEX

Oracle Application Express (APEX) stellt eine Entwicklungs- und Laufzeitumgebung zur Erstellung und Ausführung von Web-Anwendungen dar, von denen die Hauptfunktionalität aus der Bearbeitung und Darstellung von Daten aus einer Datenbank besteht. APEX ist ein integrierter Teil von der Oracle-Datenbank, der auch nur mit dieser Datenbank funktioniert.

APEX basiert auf dem Konzept des Rapid Application Development (RAD). Beim RAD geht es um eine schnelle Entwicklung von einem Prototyp einer Anwendung, damit dieser beim Auftraggeber bzw. Benutzer schon so früh wie möglich vorgelegt wird. So können die Anforderungen an die Anwendung durch Nutzungserfahrungen besser festgelegt werden.

Bei APEX werden die Web-Anwendungen als HTML-Dokumente in der Datenbank dynamisch erzeugt, die von Web-Browsern aufgerufen werden können. Mit APEX können Web-Anwendungen auf der Bedienoberfläche einfach per Mausklick erstellt werden, ohne dass das Schreiben oder die Zusammenstellung von Quellcodes notwendig ist. APEX enthält auch verschiedene Programm-Assistenten, die die Erstellung und Konfiguration von Web-Anwendungen vereinfachen. Somit können Web-Anwendungen mit geringen Programmier-

kenntnissen entwickelt werden. Eine solche Art von Anwendungsentwicklung und Programmierung wird als deklarativ bezeichnet.

APEX ist selbst eine webbasierte Anwendung, deren Benutzeroberfläche in einem Web-Browser dargestellt wird. Die folgende Abbildung zeigt die Hauptseite der Benutzeroberfläche von APEX, die von der Version Oracle Database 11g Express Edition mit der APEX-Version 4.0.2 stammt.

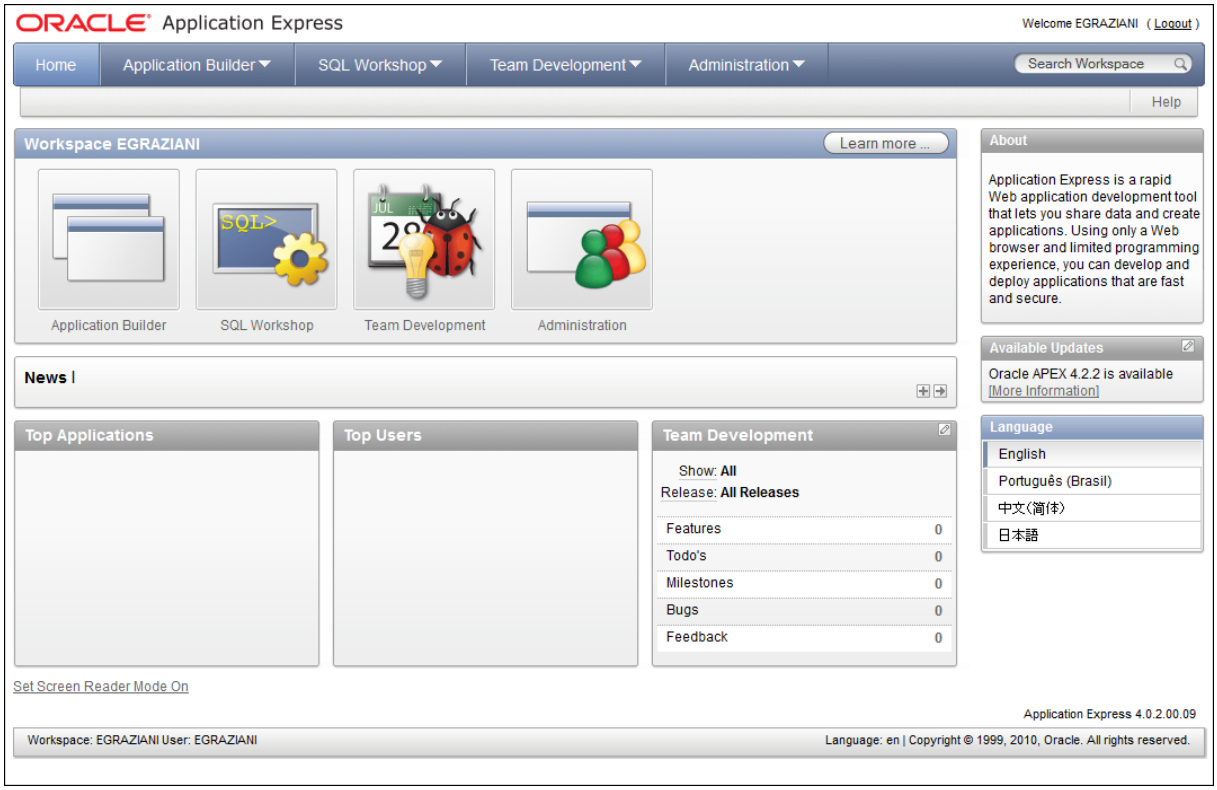

Abbildung 64: Hauptseite der Benutzeroberfläche von APEX

3.4.2 Komponenten von APEX

APEX besteht aus den Komponenten Application Builder, SQL Workshop, Team Development und Administration.

Die Komponente Application Builder dient der Erstellung von Datenbank- und Web-Anwendungen. Die Datenbank-Anwendungen (Database Applications) werden durch Pro-

gramm-Assistenten realisiert und können mehrere Seiten (Pages) enthalten, die in Regionen (Regions) aufgeteilt sind. Diese Regionen einer Seite können aus verschiedenen Elementen bestehen, wie z. B. Texte, benutzerdefinierte PL/SQL-Skripte, Diagramme, Karten, Kalender, Web-Service-Referenzen, Tabellen oder Formulare. Wie bei Datenbank-Anwendungen werden die Web-Anwendungen auch in Seiten und Regionen organisiert. Die Web-Anwendungen (Websheets) werden mit einem WYSIWYG-Editor[10] erstellt, mit dem der Inhalt der Regionen und die gesamte Struktur verwaltet und gesteuert werden können. Es ist auch möglich, Daten zu strukturieren, ohne dass SQL-Anweisungen geschrieben werden müssen. In die Web-Anwendungen, aus denen HTML-Dokumente entstehen, können Texte, Bilder, Tabellen oder Diagramme eingefügt werden. Auch Dateien können angehängt werden. Die HTML-Dokumente können auch in anderen Web-Seiten außerhalb der APEX-Umgebung verlinkt und aufgerufen werden.

SQL Workshop enthält verschiedene Tools, die das Anzeigen und die Verwaltung von Datenbank-Objekten ermöglichen. Diese Datenbank-Objekte können sowohl bei Datenbank-Anwendungen als auch Web-Anwendungen verwendet werden. In dieser Komponente können auch SQL-Anweisungen eingegeben und ausgeführt werden.

Die Komponente Team Development enthält nützliche Features, die bei der Verwaltung von Lebenszyklus einer zu entwickelnden Anwendung unterstützen. Diese Features sind für große Anwendungen geeignet, die viele Entwicklungsmitarbeiter haben.

Mit der Komponente Administration können Arbeitsbereiche (Workspace), Arbeitsbereich-Aktivität, Dienstleistungen, Benutzer und Zugriffsrechte innerhalb von APEX und Oracle verwaltet werden.

[10] *What You See Is What You Get (Echtbilddarstellung-Editor)*

3.4.3 Erstellung eines Diagramms mit APEX

In diesem Kapitel werden die wichtigen Schritte zur Erstellung eines Diagramms in APEX als Beispiel beschrieben.

Bevor ein Diagramm erstellt werden kann, muss eine Datenbank- oder Web-Anwendung in dem entsprechenden Arbeitsbereich erzeugt werden. Dafür wird die APEX-Komponente Application Builder eingesetzt, wie die folgende Abbildung darstellt.

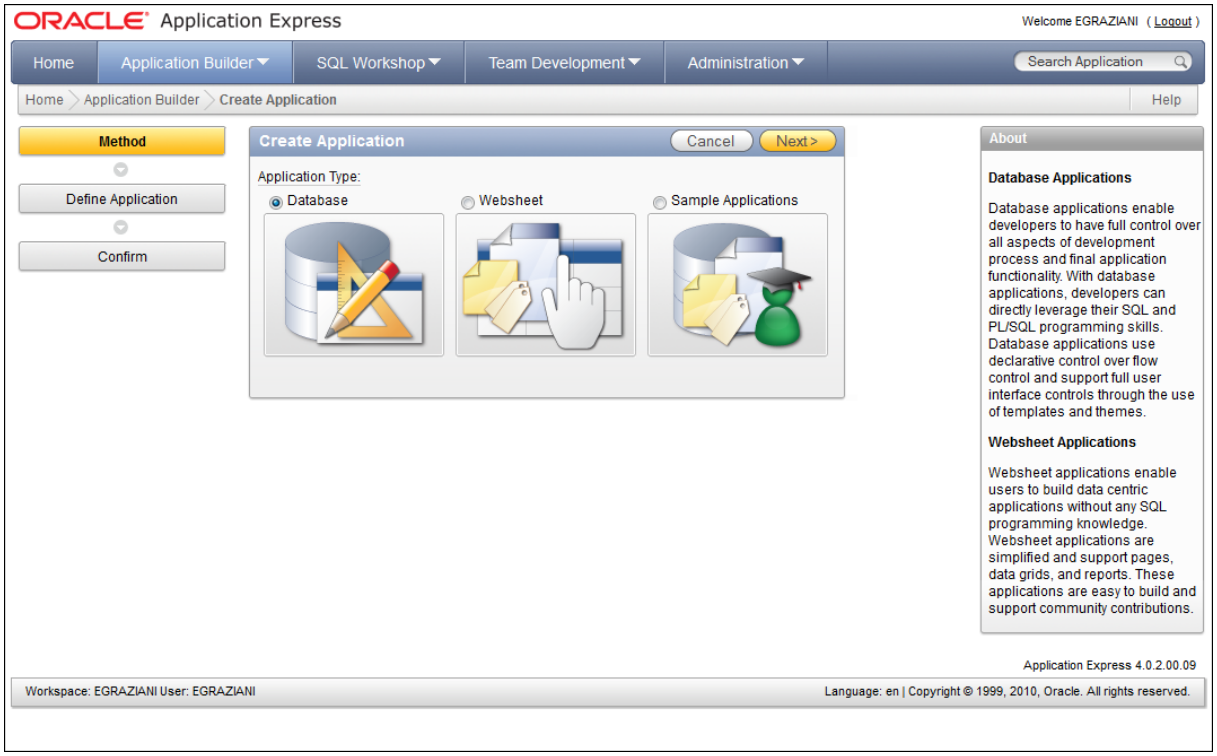

Abbildung 65: Oberfläche zur Erzeugung einer Anwendung in APEX

Bei der Erzeugung der Anwendung können verschiedene Einstellungen vorgenommen werden, unter anderem die Bestimmung des Namens der Anwendung, die Anwendungs-ID, eine Methode zur Erzeugung der Anwendung und das entsprechende Schema der Datenbank, das in der Anwendung verwendet werden soll. Die folgende Abbildung zeigt die Oberfläche von APEX, auf der diese Einstellungen enthalten sind.

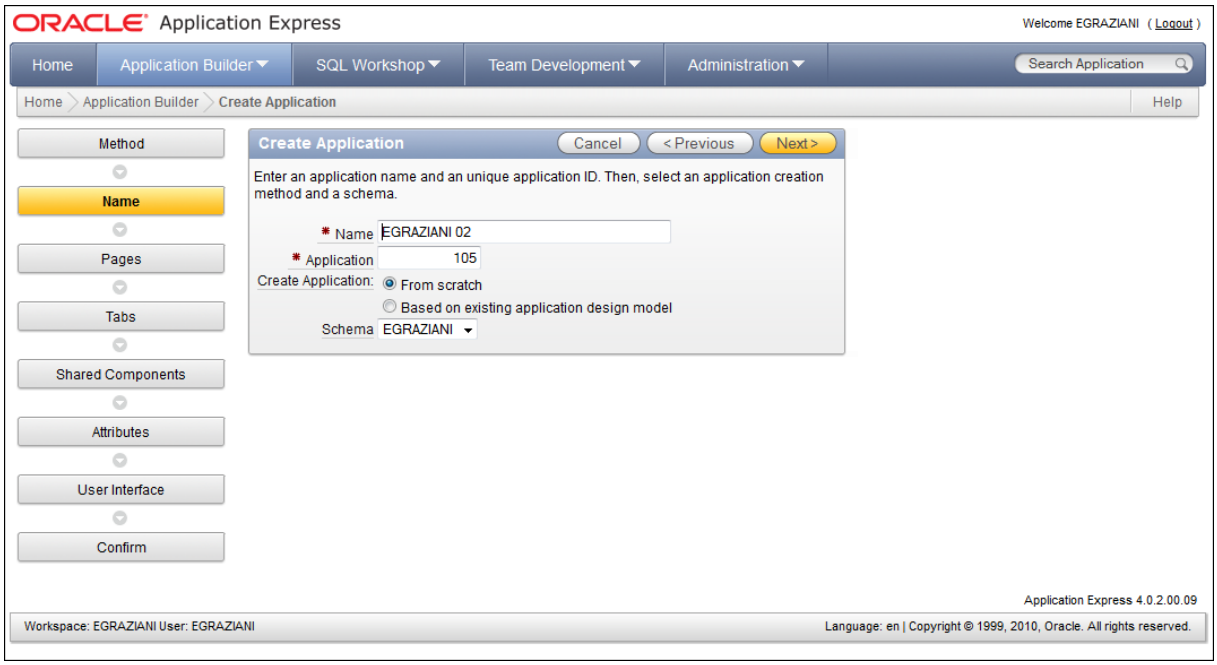

Abbildung 66: Einstellungen zur Erzeugung einer Anwendung in APEX

Nach der Einstellung der Anwendung wird eine Seite erstellt. Es gibt verschiedene Typen von Seiten, die ausgewählt werden können. Auf einer Seite können unterschiedliche Regionen gebildet werden, in denen Elemente platziert werden. Man kann aber auch bestimmen, dass eine Seite z. B. nur ein Diagramm enthalten soll. Dabei werden die Einstellungen vom Diagramm vorgenommen. Die wichtigen Einstellungen sind, von welchem Typ das Diagramm sein soll und die SELECT-Anweisung, mit der die Daten aus der Datenbank ausgelesen und in dem Diagramm dargestellt werden.

In diesem Beispiel wird ein zweidimensionales Säulendiagramm erstellt, dessen Daten aus der folgenden SELECT-Anweisung kommen.

```
SELECT datum, tool_id, anzahl_aufrufe
FROM stats_tool_aufgerufen
WHERE datum BEETWEEN '01.08.2012' AND '02.08.2012';
```

Abbildung 67: SELECT-Anweisung, die die benötigten Daten des Diagramms bezieht

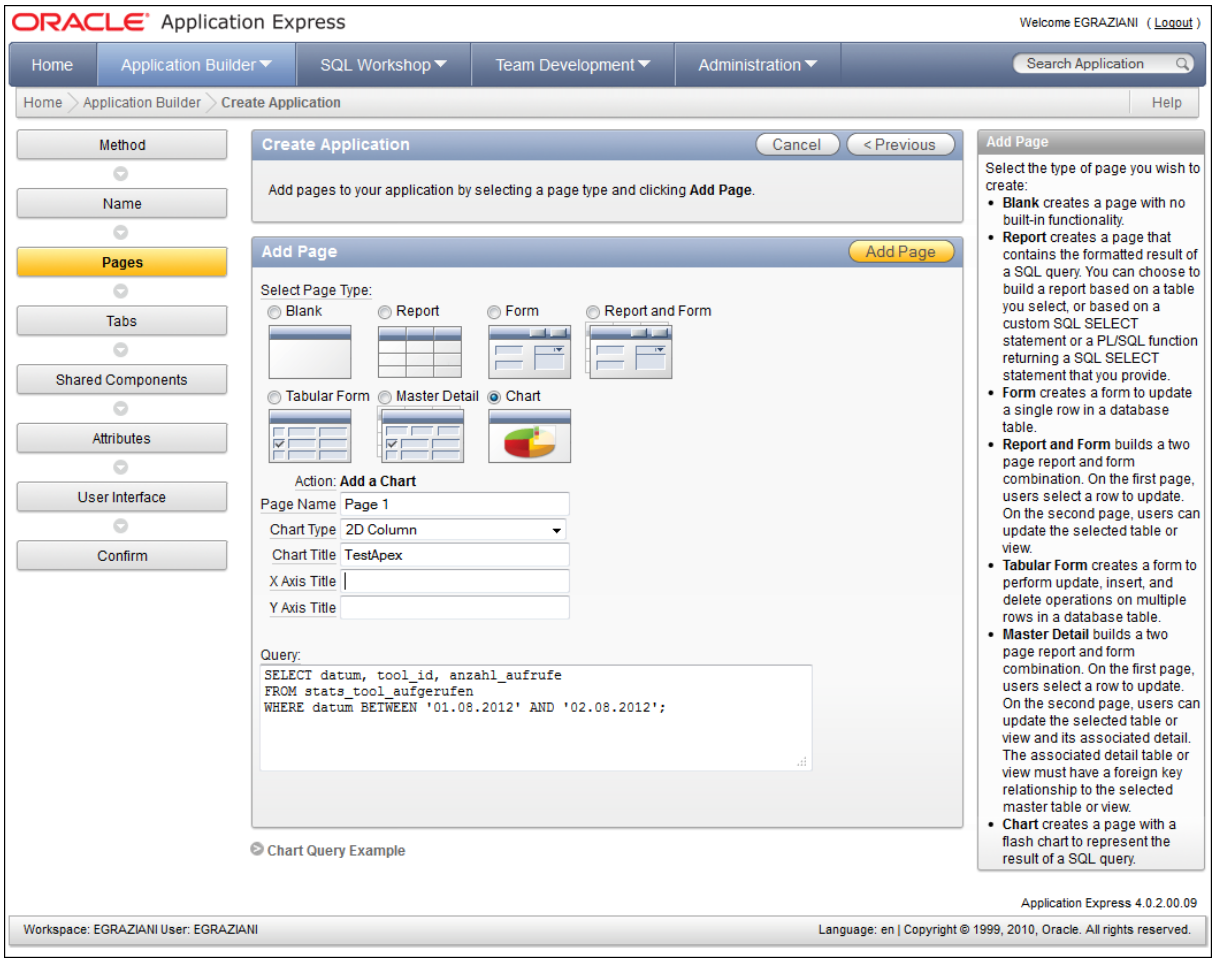

Abbildung 68: Erstellung einer Seite mit einem Diagramm in APEX

Nach der Erstellung der Seite mit dem Diagramm werden weitere Einstellungen für die Anwendung festgelegt, wie z. B. die Verwendung von Registerkarten auf der Seite, die Bestimmung von gemeinsam genutzten Komponenten, von Attributen und von einer Benutzeroberfläche. Danach muss noch die Erzeugung der Anwendung bestätigt werden.

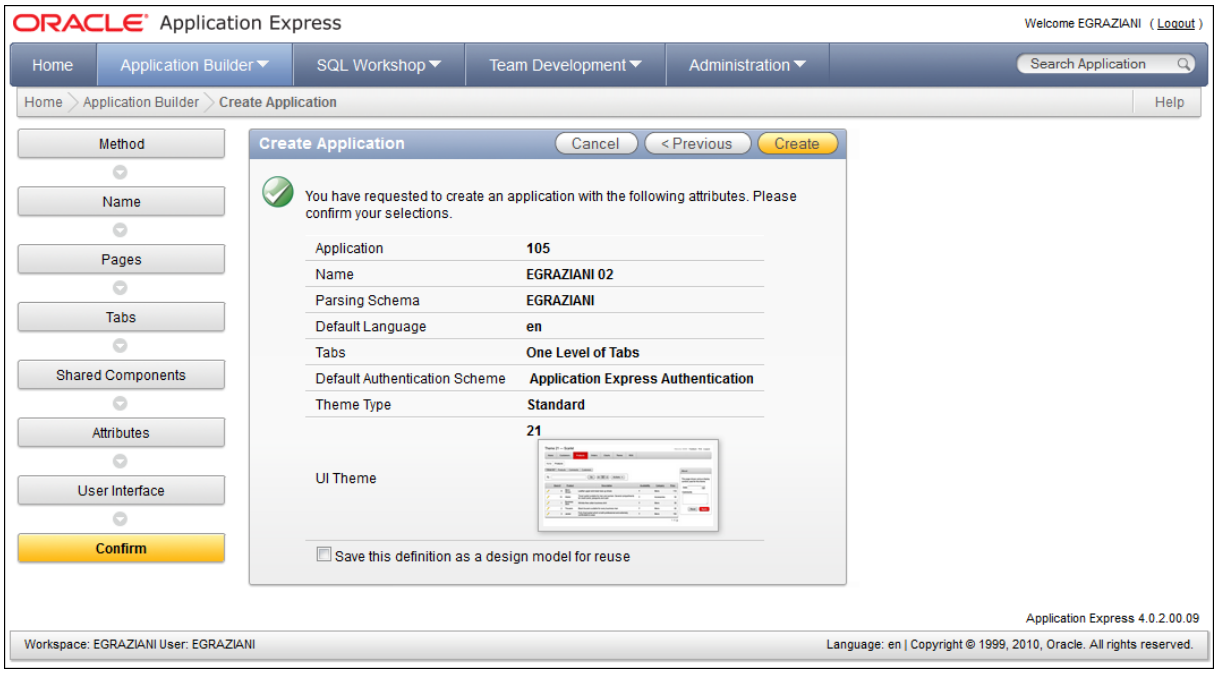

Abbildung 69: Bestätigung zur Erstellung einer Seite in APEX

Mit der Fertigstellung der Anwendung kann diese gestartet werden und die Seite aufgerufen werden, damit der Inhalt mit dem Diagramm dargestellt wird. Wie die folgende Abbildung zeigt.

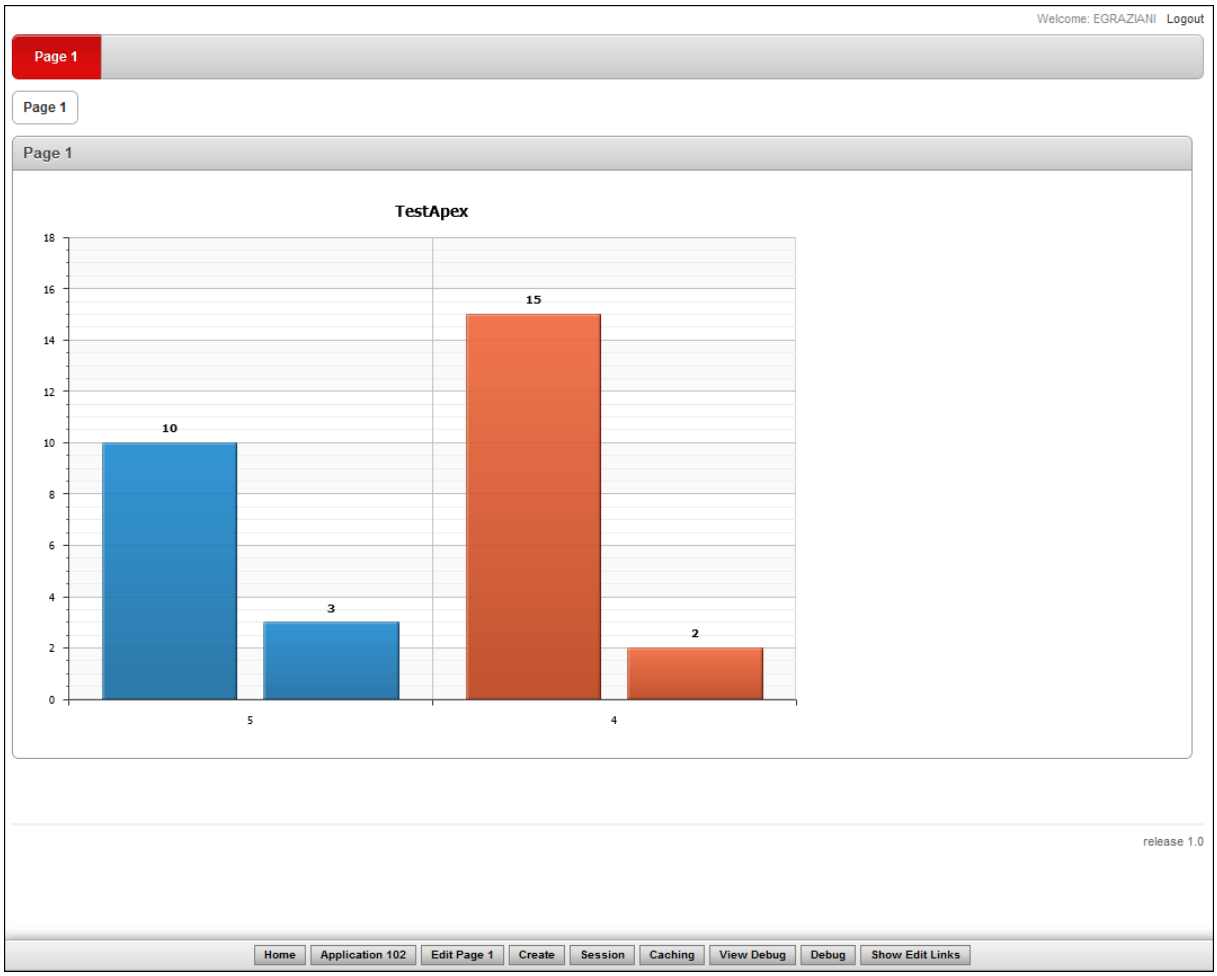

Abbildung 70: Erstelltes Diagramm in APEX

3.4.4 Erstellung eines Diagramms mit DOJO

Im Gegensatz zu APEX müssen Diagramme in Dojo implementiert werden. Dojo stellt aber verschiedene vorgefertigte Module zur Verfügung, mit denen Diagramme einfach erstellt werden können.

Zur Erstellung von Diagrammen müssen zuerst die benötigten Module geladen werden, die sich im Paket Dojox befinden. Das Paket Dojox enthält das Modul dojox/charting/Chart, mit dem das Grundgerüst eines Diagramms gebildet wird. Das Diagramm selbst wird aus Modulen des Paktes dojox/charting/plot2d festgelegt, wie z. B. das Modul dojox/charting/plot2d/Columns zur Erstellung eines zweidimensionalen Säulendiagramms. Das Paket dojox/charting enthält eine große Anzahl von Funktionen zur dynamischen Erstellung von Diagrammen. Mit dem Modul dojox/charting/axis2d/Default können die Achsen des Diagramms definiert werden. Zusätzlich können Markierungen mit dem Modul dojox/charting/plot2d/Markers und auch Themen zum Styling von Diagrammen wie dojox/charting/themes/Claro verwendet werden.

Die folgende Abbildung zeigt ein einfaches Beispiel von einem Quellcode einer HTML-Datei zur Erstellung eines Säulendiagramms mit Dojo. Nach der Deklaration der Module, die geladen werden, werden die Daten in einem Array gespeichert, die das Diagramm darstellen sollen. Danach wird das Grundgerüst des Diagramms mit dem Div-Tag erzeugt, in dem das Diagramm auf der Web-Seite dargestellt werden soll. Mit der Funktion setTheme() wird ein Thema festgesetzt. Die Funktion addPlot() definiert, welcher Typ von Diagramm erstellt wird. Das Diagramm kann in dieser Funktion auch konfiguriert werden. Mit der Funktion addAxis() werden dann die X- und Y-Achsen des Diagramms festgelegt. Danach müssen nur noch die Daten durch die Funktion addSeries() zum Diagramm hinzugefügt werden und das Diagramm durch die Funktion render() gerendert werden.

```
<!DOCTYPE html>
<html lang="de">
<head>
<meta charset="utf-8">
</head>
<body>

<div id="diagrammDivTag" style="width:400px;height:200px;"></div>

<script src="http://ajax.googleapis.com/ajax/libs/dojo/1.9.1/dojo/dojo.js"></script>

<script type="text/javascript">
        require([
           "dojox/charting/Chart",
           "dojox/charting/themes/Claro",
           "dojox/charting/plot2d/Columns",
           "dojox/charting/plot2d/Markers",
           "dojox/charting/axis2d/Default",
           "dojo/domReady!"
        ], function(Diagramm, thema) {

           var diagrammDaten = [900, 700,100];

           var diagramm = new Chart("diagrammDivTag");

           diagramm.setTheme(thema);

           diagramm.addPlot("default", {
              type: "Columns",
              markers: true,
              gap: 70
           });

           diagramm.addAxis("x", { label: true,
                                   labels: [{value: 1, text: "2011"},
                                            {value: 2, text: "2012"},
                                            {value: 3, text: "2013"}]
                                 });
           diagramm.addAxis("y", { min:0, vertical: true, fixLower: "major", fixUpper: "major"});

           diagramm.addSeries("Jahre", diagrammDaten);
           diagramm.render();
        });

</script>

</body>
</html>
```

Abbildung 71: Quellcode zur Erstellung eines Säulendiagramms in Dojo

Mit dem Aufruf der HTML-Datei wird das Diagramm angezeigt, wie die folgende Abbildung darstellt.

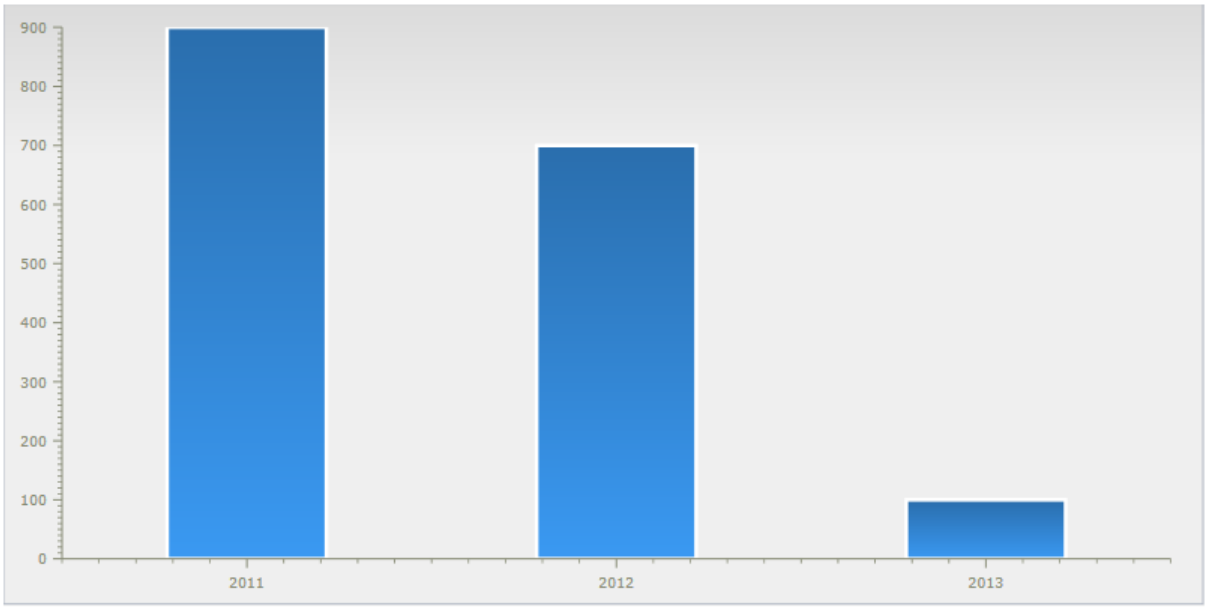

Abbildung 72: Erstelltes Diagramm in Dojo

3.4.5 Vergleich zwischen APEX und DOJO

Aus dem Vergleich zwischen APEX und DOJO ist ersichtlich, dass sich die Erstellung von Diagrammen sowohl bei APEX als auch bei Dojo einfach umsetzen lässt.

Bei der Verwendung von APEX würden die Diagramme in externe HTML-Dateien erstellt, die dann in das Statistiktool eingebaut würden. Mit APEX können aber nur Grundfunktionalitäten von einer WEB-Anwendung per Mausklick programmiert werden. Solche Art von Programmierung hat einen großen Nachteil, weil dadurch das Debuggen einer Anwendung erschwert wird. Zu weiteren und komplexeren Funktionalitäten ist die Verwendung und Implementierung von HTML, JavaScript, AJAX und CSS notwendig. Damit kann der große Vorteil von APEX, der sich durch die einfache Programmierung von Web-Anwendungen und die Erstellung von Diagrammen per Mausklick auszeichnet, nicht wirklich angewendet werden. Der Grund dafür ist, dass das Statistiktool einige komplexe Funktionalitäten enthalten muss und die Diagramme bestimmte Eigenschaften besitzen müssen, die auch einfach verändert und angepasst werden sollen.

Bei Dojo müssen die Diagramme im Statistiktool implementiert werden. Daraus folgt, dass mehr Quellcode geschrieben werden muss. Das hat aber den Vorteil, dass der Zusammenhang und die Verbindung zwischen den Diagrammen und allen Elementen im Statistiktool verständlicher werden und dadurch die Veränderungen und Anpassungen dieser Elemente vereinfacht werden. Damit kommt man zur Erkenntnis, dass die Erstellung der Diagramme im Statistiktool bei Dojo deutlich flexibler als bei APEX ist. Dojo stellt eine Vielzahl an vorgefertigten Widgets und Modulen zur Verfügung, die die Implementierung der Diagramme sehr leicht machen, wie das Beispiel in der Abbildung 71 „Quellcode zur Erstellung eines Säulendiagramms in Dojo“ zeigt. Die Widgets von Dojo können einfach konfiguriert werden, indem deren Eigenschaften durch die Wertzuweisungen in Attribute verändert und definiert werden können. Dojo enthält auch vorgefertigte CSS-Themen und Animationen, die in den Diagrammen angewendet werden können, um das allgemeine Styling der Diagramme zu bestimmen. Es ist auch möglich, vorgefertigte Tooltips und Hover-Effekte zum Diagramm hinzuzufügen.

Alle diese Gründe sprechen für die Verwendung von Dojo Toolkit, das auch im Statistiktool eingesetzt wurde.

4. Realisierung

In diesem Kapitel wird die Realisierung vom Konzept des Statistiktools aus dem Kapitel „3. Konzeption“ beschrieben.

Zuerst werden die Entwicklungswerkzeuge und Techniken kurz erläutert, die zur Entwicklungsumgebung gehören und in der Umsetzung des Konzepts eingesetzt wurden. Der Schwerpunkt dieses Kapitel liegt aber an der Implementierung des Statistiktools. Zum Kapitel „4.2 Implementierung des Statistiktools“ gehören die entwickelten Komponenten des Statistiktools sowie die Gestaltung der Benutzeroberfläche und die Erstellung der Statistik-Tabellen mit den SELECT-Anweisungen, mit denen die Statistik-Daten berechnet werden

4.1 Verwendete Entwicklungswerkzeuge und Techniken

4.1.1 Eclipse

Eclipse ist eine Entwicklungsumgebung für verschiedene Arten von Software, die von der Eclipse Foundation als Open-Source zur Verfügung gestellt wird. Die Version Eclipse Java EE IDE for Web Developers enthält alle benötigten Funktionen, mit denen die Implementierung von Web-Anwendungen mit JSPs und Servlets unterstützt wird. Diese Version von Eclipse wurde zur Entwicklung des Statistiktools verwendet.

4.1.2 Apache Tomcat Server

Der Apache Tomcat Server besteht aus einem kompletten HTTP-Server und einem Servlet-Container. Mit dem Servlet-Container und einem JSP-Compiler steht eine Umgebung zur Ausführung von Java-Code auf Web-Servern zur Verfügung.

Der Apache Tomcat Server Version 6 wurde in die Entwicklungsumgebung Eclipse eingebunden, der für die Entwicklung des Statistiktools notwendig war, weil dieser auf dem ADVBS06-Server installiert ist und somit zur Produktionsplattform gehört.

4.1.3 Oracle DB und Oracle SQL-Developer

Die Oracle-Datenbank stellt eine relationale Datenbank mit Objekten und Extensible Markup Language(XML)-Funktionen dar, die von der Firma Oracle Corporation entwickelt und vertrieben wird. Bei einer relationalen Datenbank erfolgt die Speicherung von Daten in zweidimensionalen Tabellen, die aus Zeilen und Spalten bestehen.

Auf der Produktionsplattform (ADVBS06-Server) wird ein Oracle-Datenbank-System ausgeführt, von dem alle Anwendungen auf dem EDB-Portal die benötigten Daten laden können. Das Statistiktool greift auch auf diese Datenbank zu, um Daten zu laden und zu speichern.

Zur Entwicklung des Statistiktools wurde das kostenfreie Programm Oracle SQL Developer verwendet, das die Ausführung von SQL-Anweisungen und eine einfache Verwaltung von Daten der Datenbank ermöglicht. Damit die Java-Klassen (Servlets) auf die Datenbank zugreifen können, muss ein entsprechender Oracle-JDBC-Treiber in die Entwicklungsumgebung eingebunden werden, der von Java-Klassen geladen wird.

4.1.4 SQL

Structured Query Language (SQL) ist eine auf der relationalen Algebra basierenden Programmiersprache, die von der Firma IBM entwickelt wurde. Die Syntax von SQL ist einfach aufgebaut und sie enthält unterschiedliche vorgefertigte Funktionen, die aufgerufen werden können.

SQL kann in verschiedenen relationalen Datenbanken eingesetzt werden, mit der Datenstrukturen in die Datenbanken erstellt werden können. Mit SQL können Daten aus unterschiedli-

chen Datentypen bearbeitet werden, indem Daten eingefügt, verändert oder gelöscht werden. Durch die Ausführung von SELECT-Anweisungen können Daten der Datenbank gelesen und ausgegeben werden.

4.1.5 Java

Java ist eine objektorientierte Programmiersprache, die von Sun Microsystems entwickelt wird und zur Implementierung von unterschiedliche Arte von Software eingesetzt werden kann. Mit der Java-EE-Technologie bietet Java die Möglichkeit an, Web-Anwendungen durch JSP und Servlets zu programmieren.

Die Programmiersprache Java gehört zur Java-Technologie, die auch das Java-Entwicklungswerkzeug (JDK) und die Java-Laufzeitumgebung (JRE) enthält. Die JRE besteht selbst aus der virtuellen Maschine (JVM) und den Bibliotheken. Die Verwendung von der JVM ist ein wichtiger Vorteil von Java, weil plattformunabhängige Programme implementiert werden können. Damit können Programme ohne Änderungen auf verschiedenen Betriebssystemen ausgeführt werden, solange eine entsprechende JRE auf dem Rechner installiert ist.

4.1.6 HTML

Hypertext Markup Language (HTML) ist eine Auszeichnungssprache, mit der der Inhalt eines Dokuments strukturiert werden kann.

Die Struktur eines HTML-Dokuments wird durch verschiedene Tags abgebildet, die die Elemente des Dokuments einschließen. HTML-Dokumente bestehen aus einem Head-Teil, der Metainformationen über das Dokument beinhaltet und einem Body-Teil, in dem der Inhalt des Dokuments dargestellt wird.

Zum Inhalt eines HTML-Dokuments gehören verschiedene Elemente, wie Texte, Grafiken und Hyperlinks, die zu anderen HTML-Dokumenten oder anderen Ressourcen, wie Animationen, Video- oder Audio-Dateien verweisen können.

Ein HTML-Dokument wird von einem Server zu einem Client geschickt, nachdem es von einem Client angefordert wurde. Auf den Clients werden HTML-Dokumente durch einen Web-Browser dargestellt.

4.1.7 CSS

Cascading Style Sheets (CSS) ist eine Formatierungssprache, mit der die Gestaltung und die Darstellung von HTML-Dokumenten definiert werden. Mit CSS können unter anderem Layouts, Farben, Schrifteigenschaften, Größen und Positionen von Elementen festgelegt werden. Die CSS-Elemente können in das HTML-Dokument oder in eine externe CSS-Datei gespeichert werden, die in das HTML-Dokument eingebunden wird.

CSS wurde entwickelt, damit die inhaltliche Struktur des HTML-Dokuments von dessen konkreter Darstellung getrennt bearbeitet werden kann.

4.1.8 JavaScript

JavaScript ist eine für das Internet entwickelte Skriptsprache, die eine dynamische Gestaltung von HTML in Web-Browser ermöglicht. Mit JavaScript können Web-Seiten Aktionen von Benutzer empfangen und entsprechend reagieren, indem Elemente des HTML-Dokuments verändert, nachgeladen oder erzeugt werden.

Ein JavaScript-Code kann in Script-Tags von HTML-Dokumenten platziert oder in externe JavaScript-Dateien gespeichert werden, die beim Aufruf des HTML-Dokuments geladen werden.

4.1.9 AJAX

Asynchronous JavaScript and XML (Ajax) stellt ein Konzept für einen asynchronen Datenaustausch zwischen Server und Web-Browser dar. Mit Ajax sollen Elemente einer Web-Seite verändert bzw. geladen werden, ohne dass die ganze Web-Seite neu geladen werden muss.

Bei Ajax werden Daten und Inhalt von Web-Seiten durch das DOM-Objekt repräsentiert. Mit JavaScript kann das DOM-Objekt manipuliert werden. JavaScript wird auch verwendet, um eine dynamische Darstellung von Elementen auf Web-Seiten zu ermöglichen.

Ein wichtiger Bestandteil von Ajax ist das XMLHttpRequest-Objekt, das in vielen Web-Browsern implementiert ist. Mit diesem Objekt können Daten asynchron übertragen werden. Die Kommunikation zwischen Server und Web-Browser werden durch HTTP-Anfragen realisiert.

Die Daten können bei Ajax in XML-Format übertragen werden. XML (Extensible Markup Language) ist eine Auszeichnungssprache, die die Daten in eine hierarchisch strukturierte Form darstellt. Es gibt aber auch andere Formate, wie z. B. JSON, die zur Übertragung der Daten auch verwendet werden können.

4.1.10 JSON

JSON (JavaScript Object Notation) ist ein einfaches und kompaktes Datenformat, mit dem Daten zwischen Anwendungen übertragen werden können. JSON stellt ein für Menschen einfach lesbares und schreibbares Textformat dar und ist unabhängig von Programmiersprachen. Verschiedene Programmiersprachen, wie C++, C#, Java, JavaScript, Perl, Python und PHP, besitzen entsprechende Methoden zum Parsen und Generieren von JSON-Objekten.

JSON kann als Ersatz für XML in Ajax-Anwendungen verwendet werden. Der große Vorteil von JSON gegenüber dem XML-Format liegt darin, dass kleinere Dateien erzeugt werden,

weil JSON nicht so viele Tags wie XML braucht, sodass die Datenübertragung schneller durchgeführt werden kann.

In der Darstellung eines JSON-Objekts werden dessen Daten immer von zwei geschweiften Klammern eingeschlossen. Die innere Struktur des Objekts kann zwei unterschiedliche Typen von Daten enthalten. Der erste Typ besteht aus einem Namen und einem Wert, die durch einen Doppelpunkt getrennt werden. Der zweite Typ ist eine geordnete Wertliste, die wie ein Array realisiert wird, indem die Daten von zwei eckigen Klammern eingeschlossen werden. Die folgende Abbildung zeigt ein Bespiel von der Struktur eines JSON-Objekts.

```
{
        "Hochschule" : "Fachhochschule Köln, Campus Gummersbach",
        "Adresse" : {
                "Straße" : "Steinmüllerallee",
                "Hausnummer" : 1,
                "PLZ" : 51643,
                "Ort" : "Gummersbach"
        },
        "Telefon" : [
                "02261 / 8196-0",
                "02261 / 8196 -6840",
                "02261 / 8196 -6666"
        ]
}
```

Abbildung 73: Beispiel von der Struktur eines JSON-Objekts

4.2 Implementierung des Statistiktools

Die Struktur des Statistiktools wurde in drei Schichten aufgebaut, die aus der Persistenz-, Anwendungs- und Präsentationsschicht besteht. In der Persistenzschicht soll das Statistiktool ermöglicht werden, die Statistik-Daten dauerhaft zu speichern, damit diese nicht immer wieder erzeugt werden müssen. In der Anwendungsschicht sind alle Komponenten, die der Durchführung von Anwendungslogik-Aufgaben dienen. Die Präsentationsschicht enthält Komponenten zur Darstellung von der Benutzeroberfläche des Statistiktools.

Die Implementierung des Statistiktools wird in eine Modell 1-Architektur vom Entwicklungsmodell bei JSP-Anwendungen umgesetzt, wie die Abbildung 74 „Architektur des Statistiktools“ zeigt.

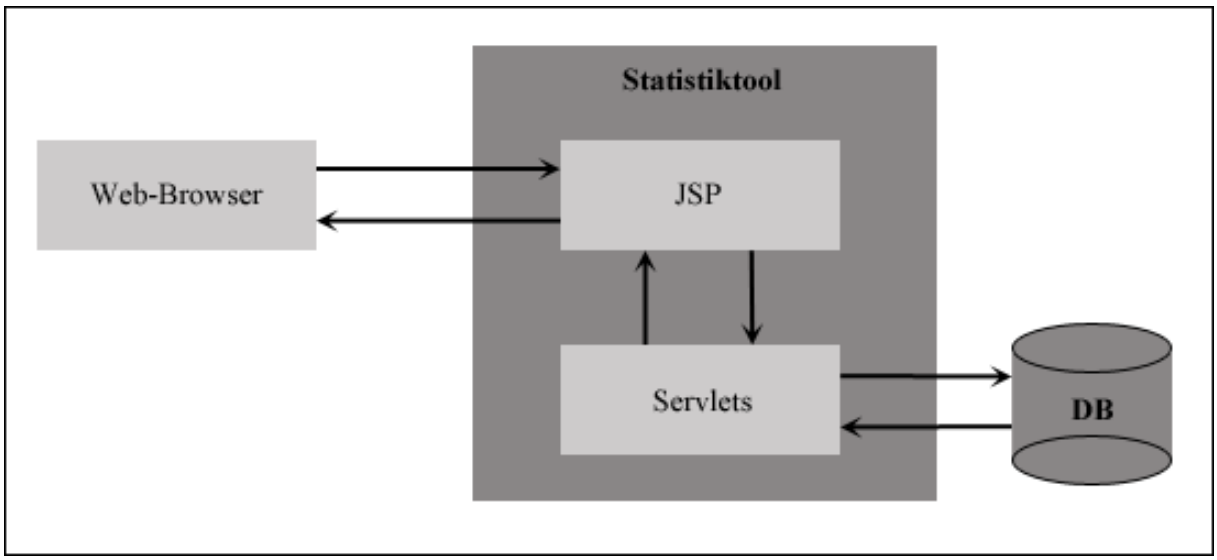

Abbildung 74: Architektur des Statistiktools

Bei der Implementierung des Statistiktools war der erste Schritt, die Persistenzschicht zu realisieren, indem die Statistik-Tabellen im Statistiktool-Schema erstellt wurden. Dafür wurde ein SQL-Skript geschrieben, das alle Statistik-Tabellen anlegt. Mit den Statistik-Tabellen konnten dann schon die SELECT-Anweisungen programmiert werden, die die Statistik-Daten berechnen. Danach konnten einige Tests durchgeführt werden, um zu überprüfen, ob die Statistik-Daten richtig berechnet werden.

Der zweite Schritt war die Realisierung der Anwendungs- und der Präsentationsschicht, indem die einzelnen Komponenten des Statistiktools implementiert wurden. Die Hauptkomponente der Präsentationsschicht ist eine JSP-Datei, die die Benutzeroberfläche des Statistiktools darstellt und für die Kommunikation mit dem Web-Browser des Clients zuständig ist. Diese JSP-Datei leitet auch die Abfragen vom Client an die Servlets weiter. Die Servlets sind dagegen für die Anwendungslogik des Statistiktools verantwortlich und befinden sich in der Anwendungsschicht. Die Servlets empfangen die gesendeten Abfragen und holen die angeforderten Daten aus dem Statistiktool-Schema, die an die JSP-Datei gesendet werden. Die JSP-Datei erstellt mit den gesendeten Daten die Diagramme, die auf der Benutzeroberfläche dargestellt werden. Die Erstellung der Diagramme wird von Funktionen von implementierten Modulen durchgeführt, die von Dojo-Loader in JSP mitgeladen werden. Es gibt auch ein Servlet, das nur für die Aktualisierung der Statistik-Tabellen verantwortlich ist.

In den folgenden Kapiteln werden die Statistik-Tabellen, die Implementierung der Komponenten und die Gestaltung der Benutzeroberfläche des Statistiktools beschrieben.

4.2.1 Persistenzschicht

4.2.1.1 Beschreibungen der Statistik-Tabellen

Die Statistik-Tabellen gehören zum Lösungsansatz des Performance-Problems, das im Kapitel 3.3.4.2 beschrieben wurde. Aus den 18 Tabellen des Statistiktool-Schemas, die es bereits gab, wurden zusätzlich 14 Statistik-Tabellen erstellt, um die Statistik-Daten zu speichern.

Um die Verständlichkeit zu erhöhen und die Zusammenhänge zwischen den Tabellen zu verdeutlichen, wurden die Namen der Statistik-Tabellen so ausgewählt, dass sie den Namen der entsprechenden Tabellen ähneln. Dazu wurde der Präfix „STATS“ in den Namen aller Statistik-Tabellen. So wurde z. B. für die Tabelle NATION_AUFGERUFEN die Statistik-Tabelle STATS_NATION_AUFGERUFEN angelegt.

Es war möglich, die gesammelten Datensätze einiger Tabellen zusammenzufassen und nur in eine Statistik-Tabelle zu speichern. Ein Beispiel davon sind die Tabellen TOOL_AUFGERUFEN und TOOL_BEENDET, deren gesammelte Daten berechnet und in die Statistik-Tabelle STATS_TOOL_AUFGERUFEN abgelegt werden. Bei anderen Tabellen musste der Inhalt aber getrennt werden, wie z. B. für den Inhalt der Tabelle SQL_OPTIMIZER_SCHEMA. Für diese Tabelle wurden die Statistik-Tabellen STATS_SQL_OPTIMIZER_SCHEMA und STATS_SQL_OPTIMIZER_FRAGEN erstellt. Der Grund dafür ist, dass bei der Tabelle SQL_OPTIMIZER_SCHEMA die Anzahl von Aufrufen sowohl für die Spalte SCHEMA_ID als auch für die Spalte AUFGABE_ID berechnet werden kann. Es können aber nicht beide Informationen in eine einzige Zeile gespeichert werden, weil die Datensätze bei der Berechnung nach der Spalte DATUM und noch nach der Spalte SCHEMA_ID oder AUFGABE_ID gruppiert werden. Somit musste die Anzahl der Aufrufe von einem Datenbankschema und einer Aufgabe in verschiedene Statistik-Tabellen gespeichert werden.

Jeder Statistik-Tabelle wird eine SELECT-Anweisung im Statistiktool zugeordnet, die die Statistik-Daten berechnet. Eine wichtige Eigenschaft dieser SELECT-Anweisungen besteht aus der Gruppierung der gesammelten Daten. Dafür werden bei allen Statistik-Tabellen eine ID-Spalte, wie z. B.AUFGABE_ID oder SCHEMA_ID, und die Spalte DATUM verwendet. Aber damit das Datum zur Gruppenbildung in der SELECT-Anweisung verwendet werden kann, muss der Inhalt der Spalte DATUM zuerst mit der Funktion TO_CHAR() konvertiert werden. Das Problem liegt darin, dass der Datentyp DATE die Einheiten Jahr, Monat, Tag, Stunden, Minuten und Sekunden enthält und dadurch ein Datumvergleich zur Gruppierung nicht möglich ist, weil die Gleichheit der Daten nur erreicht ist, wenn alle Einheiten auch gleich sind. Deswegen werden nur die Einheiten Tag, Monat und Jahr aus dem Datum mit der Funktion TO_CHAR(DATUM, 'DD.MM.YYYY') entnommen, in einen String umgewandelt und verglichen.

Im Folgenden werden einzelne Statistik-Tabellen mit ihren Strukturen, Inhalt und den entsprechenden SELECT-Anweisungen zur Berechnung der Statistik-Daten beschrieben.

4.2.1.1.1 Tabelle STATS_AKT_STAND

Die Statistik-Tabelle STATS_AKT_STAND enthält den aktuellen Stand der Statistik und keine Statistik-Daten. Aus dieser Tabelle können die Datumsangaben entnommen werden, anhand derer bestimmt wird, seit wann Daten für die Statistik gesammelt werden und wann die letzte Berechnung der gesammelten Daten durchgeführt wurde. Diese Informationen werden in die Spalten VON und BIS gespeichert. Der daraus entstandene Zeitraum gibt die vorhandene Statistik im Statistiktool-Schema an, die vom Statistiktool abgefragt werden kann.

Bei jeder neuen Berechnung der gesammelten Daten muss der Wert der Spalte BIS mit einer UPDATE-Anweisung aktualisiert werden, während der Wert der Spalte VON immer gleich bleibt. Falls noch keine Statistik berechnet wurde, haben die Spalten VON und BIS den Wert NULL.

Die Spalte TOOL ist der Primärschlüssel der Tabelle und enthält den Namen der Tools bzw. Trainer. Zurzeit beinhaltet die Spalte TOOL die Werte „alle“ und „SQL-Abfrageoptimierungstool“. Es war notwendig, die Tools in der Tabelle zu trennen, weil sich die Werte in der Spalte VON unterscheiden. Der Grund dafür ist, dass das SQL-Abfrageoptimierungstool erst seit 2013 auf dem EDB-Portal zur Verfügung steht und infolgedessen auch erst seit 2013 Daten für die Statistik gesammelt werden. Für alle anderen Tools und Trainer werden Daten seit dem 02.04.2012 gesammelt. Falls ein neuer Trainer auf das EDB-Portal hinzugefügt wird, ist es nötig, diesen in die Tabelle einzutragen.

Die folgenden Abbildungen zeigen die Struktur und den Inhalt der Statistik-Tabelle STATS_AKT_STAND.

```
STATS_AKT_STAND (
        TOOL            VARCHAR2(50),
        VON             DATE,
        BIS             DATE
);
```

Abbildung 75: Struktur der Tabelle STATS_AKT_STAND

TOOL	VON	BIS
alle	02.04.12	01.08.13
SQL-Abfrageoptimierungstool	20.01.13	01.08.13

Abbildung 76: Inhalt der Tabelle STATS_AKT_STAND

(Stand: 01.08.2013, 19:00 Uhr)

4.2.1.1.2 Tabelle STATS_TOOL_AUFGERUFEN

Die Statistik-Tabelle STATS_TOOL_AUFGERUFEN enthält Statistik-Daten über die Verwendung von allen Tools und Trainer des EDB-Portals. Die Spalten DATUM und TOOL_ID bilden den Primärschlüssel der Tabellen. Diese beiden Spalten dienen der Eindeutigkeit der Daten und werden im Statistiktool in die WHERE-Klausel der SELECT-Anweisung eingesetzt, um die Werte der anderen Spalten ANZAHL_AUFRUFE und ANZAHL_BEENDET abzufragen. Die Statistik-Daten, die aus den gesammelten Daten berechnet werden, sind in den Spalten ANZAHL_AUFRUFE und ANZAHL_BEENDET gespeichert.

Aus dieser Tabelle kann die Anzahl von Aufrufen eines bestimmten Trainers innerhalb eines bestimmten Tages entnommen werden. Dazu wird auch erfasst, wie oft ein bestimmter Trainer innerhalb eines bestimmten Tages bis zu Ende verwendet wurde. Das Beenden eines Trainers kann nur bei MCT, SQL-Trainer, SQL-Trainer2 und XQuery bestimmt werden. Die folgende Abbildung zeigt die Struktur Tabelle STATS_TOOL_AUFGERUFEN.

```
STATS_TOOL_AUFGERUFEN (
        DATUM                   DATE,
        TOOL_ID                 NUMBER,
        ANZAHL_AUFRUFE          NUMBER,
        ANZAHL_BEENDET          NUMBER
);
```

Abbildung 77: Struktur der Tabelle STATS_TOOL_AUFGERUFEN

Zur Berechnung der Statistik-Daten wird die folgende SELECT-Anweisung im Statistiktool ausgeführt. Diese verschachtelte SELECT-Anweisung greift auf die gesammelten Daten aus

den Tabellen TOOL_AUFGERUFEN und TOOL_BEENDET zu. Die gesammelten Daten werden nach den Spalten DATUM und TOOL_ID gruppiert und die Berechnung der Statistik-Daten für die Spalten ANZAHL_AUFRUFE und ANZAHL_BEENDET erfolgt durch die Funktion COUNT(). Diese SELECT-Anweisung berücksichtigt nur gesammelte Daten, deren Datum beispielsweise größer gleich 01.01.2013 ist.

```
SELECT TO_CHAR(ta.datum, 'DD.MM.YYYY'),
            ta.tool_id,
            COUNT(ta.tool_id),
            (SELECT COUNT(tb.tool_id)
              FROM tool_beendet tb
              WHERE TO_CHAR(ta.datum, 'DD.MM.YYYY') = TO_CHAR(tb.datum, 'DD.MM.YYYY')
              AND ta.tool_id = tb.tool_id)
FROM tool_aufgerufen ta
WHERE ta.datum >= TO_DATE('01.01.2013', 'DD.MM.YYYY')
GROUP BY TO_CHAR(ta.datum, 'DD.MM.YYYY'), ta.tool_id;
```

Abbildung 78: SELECT-Anweisung zur Tabelle STATS_TOOL_AUFGERUFEN

DATUM	TOOL_ID	ANZAHL_AUFRUFE	ANZAHL_BEENDET
04.04.12	7	1	0
04.04.12	10	1	0
10.04.12	6	46	0
11.04.12	4	26	6
19.04.12	11	2	0
22.04.12	5	3	0
22.04.12	12	1	0
25.04.12	4	33	8
25.04.12	10	1	0
26.04.12	11	3	0
27.04.12	3	25	0
28.04.12	3	17	0
29.04.12	6	193	0
30.04.12	2	4	2
01.05.12	6	234	0

Abbildung 79: Teilinhalt der Tabelle STATS_TOOL_AUFGERUFEN

(Stand: 01.08.2013, 19:00 Uhr)

4.2.1.1.3 Tabelle STATS_MCT

Die Statistik-Tabelle STATS_MCT enthält Statistik-Daten nur vom MCT und stellt die größte Statistik-Tabelle des Schemas, weil sie die meisten Möglichkeiten bietet, Statistik-Daten zu berechnen. Alle Statistik-Daten sind auf Fragen bzw. Aufgaben des MCT bezogen. Aus diesem Grund wurden die Spalten DATUM und FRAGE_ID als Primärschlüssel gesetzt.

Zu der Tabelle gehören noch sechs Spalten. Die Werte der Spalten ANZAHL_AUFRUFE, DURCHSCHNITTSZEIT_ANTWORT, ANZAHL_RICHTIG_BEANTWORTET und ANZAHL_FALSCH_BEANTWORTET werden aus den gesammelten Daten der Tabelle FRAGENAUSWERTUNG berechnet.

In die Spalte ANZAHL_AUFRUFE wird die Anzahl von Aufrufen einer bestimmten Frage im MCT gespeichert, die der Summe der Spalten ANZAHL_RICHTIG_BEANTWORTET und ANZAHL_FALSCH_BEANTWORTET entspricht.

Die Spalte DURCHSCHNITTSZEIT_ANTWORT enthält die durchschnittliche Zeit, die die Benutzer gebraucht haben, um eine Frage zu beantworten. Die längste oder die kürzeste Zeit wurden nicht mitgespeichert, weil sie durch Umstände, wie das Verhalten der Benutzer, beeinflusst bzw. verfälscht werden könnten. Sinnvoller erscheint es, die durchschnittliche Zeit zu berechnen und zu speichern, da bei der durchschnittlichen Zeit die Einflüsse ausgeglichen werden.

Die Überprüfung, wie oft eine Frage von Studenten richtig oder falsch beantwortet wurde, stellt die wichtigste Information für die Professoren dar, die aus dem MCT entnommen werden kann. Diese Information wird in die Spalten ANZAHL_RICHTIG_BEANTWORTET und ANZAHL_FALSCH_BEANTWORTET gespeichert.

Die Datensätze von den Spalten ANZAHL_AUFRUFE_FEEDBACK und ANZAHL_AUFRUFE_HILFE werden aus den Tabellen MCT_FEEDBACK bzw. MCT_HILFE berechnet und sie geben an, wie häufig das Feedback und die Hilfe einer Frage im MCT aufgerufen wurden.

```
STATS_MCT (
        DATUM                                   DATE,
        FRAGE_ID                                NUMBER,
        ANZAHL_AUFRUFE                          NUMBER,
        DURCHSCHNITTSZEIT_ANTWORT               NUMBER,
        ANZAHL_RICHTIG_BEANTWORTET              NUMBER,
        ANZAHL_FALSCH_BEANTWORTET               NUMBER,
        ANZAHL_AUFRUFE_FEEDBACK                 NUMBER,
        ANZAHL_AUFRUFE_HILFE                    NUMBER
);
```

Abbildung 80: Struktur der Tabelle STATS_MCT

Zur Berechnung der Statistik-Daten für die Statistik-Tabelle STATS_MCT wird die folgende SELECT-Anweisung aus dem Statistiktool verwendet. Der Zugriff der SELECT-Anweisung erfolgt meistens auf die Tabelle FRAGENAUSWERTUNG. Die SELECT-Anweisung enthält aber auch zwei verschachtelte SELECT-Anweisungen, die auf die Tabellen MCT_HILFE und MCT_FEEDBACK zugreifen.

Die Berechnung der Daten wird meistens mit der Funktion COUNT() durchgeführt. Bei der durchschnittlichen Antwortzeit wird die Funktion AVG() verwendet. Die Spalten ANZAHL_RICHTIG_BEANTWORTET und ANZAHL_FALSCH_BEANTWORTET werden mit der Funktion SUM() in Verbindung mit dem CASE-Ausdruck berechnet, die die Daten aus der Spalte RICHTIG_BEANTWORTET der Tabelle FRAGENAUSWERTUNG zur Berechnung verwendet. Diese Spalte kann nur die Werte 0 für falsch beantwortet oder 1 für richtig beantwortet annehmen.

Die SELECT-Anweisung aus dem Statistiktool besitzt zwei Ausdrücke in der WHERE-Klausel. Der erste Ausdruck lautet beispielsweise „fr.DATUM >= TO_DATE('01.01.2013', 'DD.MM.YYYY')“ und bedeutet, dass die SELECT-Anweisung nur die gesammelten Daten berücksichtigt, deren Datum größer gleich 01.01.2013 ist. Der zweite Ausdruck „fr.TOOL_ID = 2“ gibt an, von welchem Tool die Daten einbezogen werden sollen. In diesem Fall ist TOOL_ID immer gleich 2, weil die Nummer 2 die ID des MCT in der Tabelle TOOLS ist. In der Tabelle FRAGENAUSWERTUNG sind auch die IDs von anderen Trainern gespeichert, wie SQL-Trainer, SQL-Trainer 2 und XQuery-Trainer. Die gesammelten Daten werden nach den Spalten DATUM und FRAGE_ID gruppiert.

```
SELECT TO_CHAR(fr.datum, 'DD.MM.YYYY'),
            fr.frage_id,
            COUNT(fr.frage_id),
            AVG(fr.zeit),
            SUM(CASE WHEN richitg_beantwortet = 1 THEN 1 ELSE 0 END),
            SUM(CASE WHEN richitg_beantwortet = 0 THEN 1 ELSE 0 END),
            (SELECT COUNT(fb.frage_id)
             FROM mct_feedback fb
             WHERE TO_CHAR(fb.datum, 'DD.MM.YYYY') = TO_CHAR(fr.datum, 'DD.MM.YYYY')
             AND fb.frage_id = fr.frage_id),
            (SELECT COUNT(h.frage_id)
             FROM mct_hilfe h
             WHERE TO_CHAR(h.datum, 'DD.MM.YYYY') = TO_CHAR(fr.datum, 'DD.MM.YYYY')
             AND h.frage_id = fr.frage_id)
FROM fragenauswertung fr
WHERE fr.datum >= TO_DATE('01.01.2013', 'DD.MM.YYYY') AND fr.tool_id = 2
GROUP BY TO_CHAR(fr.datum, 'DD.MM.YYYY'), fr.frage_id;
```

Abbildung 81: SELECT-Anweisung zur Tabelle STATS_MCT

DATUM	FRAGE_ID	ANZAHL_AUFRUFE	DURCHSCHNITTSZEIT_ANTWORT	ANZAHL_RICHTIG_BEANTWORTET	ANZAHL_FALSCH_BEANTWORTET	ANZAHL_AUFRUFE_FEEDBACK	ANZAHL_AUFRUFE_HILFE
25.04.12	90767	1	25,083	1	0	0	0
25.04.12	90769	1	12,782	0	1	0	0
25.04.12	90786	1	7,903	0	1	0	0
25.04.12	91132	1	12,462	1	0	0	0
25.04.12	91624	1	7,5449996	0	1	0	0
25.04.12	91671	1	8,752	0	1	0	0
25.04.12	92571	1	7,569	0	1	0	0
26.04.12	515	2	8,6215	1	1	0	0
26.04.12	685	1	9,925	1	0	0	0
26.04.12	1510	1	4,776	0	1	0	0
26.04.12	1656	1	6,2879996	0	1	0	0
26.04.12	1666	1	3,436	0	1	0	0
26.04.12	1744	1	24,886	0	1	0	0
26.04.12	1767	1	15,882	1	0	0	0
26.04.12	1937	1	30,561	0	1	0	0

Abbildung 82: Teilinhalt der Tabelle STATS_MCT

(Stand: 01.08.2013, 19:00 Uhr)

4.2.1.1.4 Tabelle STATS_MCT_KATEGORIEN

Die Statistik-Tabelle STATS_MCT_KATEGORIEN enthält Statistik-Daten, die Informationen über die Auswahl der Kategorien im MCT durch die Benutzer darstellen. Die Tabelle besitzt die Spalten DATUM und MCT_KAT_ID als Primärschlüssel. Der Wert der Spalte ANZAHL_AUFRUFE gibt an, wie oft eine Kategorie an einem bestimmten Tag ausgewählt wurde. Zur Berechnung dieser Spalte werden die gesammelten Daten aus der Tabelle MCT_KATEGORIEN_AUSGEWAEHLT verwendet.

```
STATS_MCT_KATEGORIEN (
        DATUM                   DATE,
        MCT_KAT_ID              NUMBER,
        ANZAHL_AUFRUFE          NUMBER
);
```

Abbildung 83: Struktur der Tabelle STATS_MCT_KATEGORIEN

Zur Berechnung der Statistik-Daten für die Tabelle STATS_MCT_KATEGORIEN wird eine einfache SELECT-Anweisung im Statistiktool verwendet. Die gesammelten Daten werden nach den Spalten DATUM und MCT_KAT_ID gruppiert und die Berechnung der Spalten ANZAHL_AUFRUFE erfolgt durch die Funktion COUNT(). Diese SELECT-Anweisung berücksichtigt nur gesammelte Daten, deren Datum beispielsweise größer gleich 01.01.2013 ist.

```
SELECT TO_CHAR(datum, 'DD.MM.YYYY'),
            mct_kat_id,
            COUNT(mct_kat_id)
FROM mct_kategorien_ausgewaehlt
WHERE datum >= TO_DATE('01.01.2013', 'DD.MM.YYYY')
GROUP BY TO_CHAR(ta.datum, 'DD.MM.YYYY'), mct_kat_id;
```

Abbildung 84: SELECT-Anweisung zur Tabelle STATS_MCT_KATEGORIEN

DATUM	MCT_KAT_ID	ANZAHL_AUFRUFE
04.04.12	7	1
04.04.12	10	1
05.04.12	5	2
07.04.12	8	2
11.04.12	1	2
11.04.12	4	3
11.04.12	9	1
12.04.12	1	8
14.04.12	3	2
14.04.12	7	2
15.04.12	9	1
19.04.12	11	4
19.04.12	7	8
21.04.12	11	1
22.04.12	12	1

Abbildung 85: Teilinhalt der Tabelle STATS_MCT_KATEGORIEN

(Stand: 01.08.2013, 19:00 Uhr)

4.2.1.1.5 Tabelle STATS_MCT_UNI

In der Statistik-Tabelle STATS_MCT_UNI werden Statistik-Daten gespeichert, die die Auswahl der Universitäten oder Fachhochschulen durch die Benutzer im MCT angeben. Dieser Auswahlvorgang im MCT erfolgt vor dem Start des Tests. Diese Tabelle hat die Spalten DATUM und MCT_UNI_ID als Primärschlüssel. Der Wert der Spalte ANZAHL_AUFRUFE zeigt, wie oft eine Universität oder Fachhochschule an einem bestimmten Tag ausgewählt wurde. Zur Berechnung dieser Spalte werden die gesammelten Daten aus der Tabelle MCT_UNI_AUSGEWAEHLT verwendet.

```
STATS_MCT_UNI (
        DATUM                   DATE,
        MCT_UNI_ID              NUMBER,
        ANZAHL_AUFRUFE          NUMBER
);
```

Abbildung 86: Struktur der Tabelle STATS_MCT_UNI

Die Berechnung der Statistik-Daten für die Tabelle STATS_MCT_UNI erfolgt durch eine einfache SELECT-Anweisung im Statistiktool, die in der folgenden Abbildung gezeigt wird. In der Berechnung wird die Funktion COUNT() verwendet. Die gesammelten Daten werden nach den Spalten DATUM und MCT_UNI_ID gruppiert und die Berechnung der Spalten ANZAHL_AUFRUFE erfolgt durch die Funktion COUNT(). Diese SELECT-Anweisung berücksichtigt nur gesammelte Daten, deren Datum beispielsweise größer gleich 01.01.2013 ist.

```
SELECT TO_CHAR(datum, 'DD.MM.YYYY'),
            mct_uni_id,
            COUNT(mct_uni_id)
FROM mct_uni_ausgewaehlt
WHERE datum >= TO_DATE('01.01.2013', 'DD.MM.YYYY')
GROUP BY TO_CHAR(datum, 'DD.MM.YYYY'), mct_uni_id;
```

Abbildung 87: SELECT-Anweisung zur Tabelle STATS_MCT_UNI

DATUM	MCT_UNI_ID	ANZAHL_AUFRUFE
03.04.12	14	1
13.04.12	15	1
23.04.12	22	4
24.04.12	18	1
25.04.12	14	2
26.04.12	14	4
29.04.12	22	5
14.05.12	22	1
19.05.12	22	6
23.05.12	22	6
01.06.12	14	10
03.06.12	17	1
04.06.12	14	5
15.06.12	14	20
16.06.12	14	19

Abbildung 88: Teilinhalt der Tabelle STATS_MCT_UNI

(Stand: 01.08.2013, 19:00 Uhr)

4.2.1.1.6 Tabelle STATS_NATION_AUFGERUFEN

Die Statistik-Tabelle STATS_NATION_AUFGERUFEN enthält Statistik-Daten, die Informationen über die Auswahl der Nation bzw. Sprache von Benutzern auf dem EDB-Portal liefern. Die Spalten DATUM und NATION_ID stellen den Primärschlüssel dar. Der Wert der Spalte ANZAHL_AUFRUFE gibt an, wie oft eine Nation bzw. Sprache von Benutzern auf dem EDB-Portal an einem bestimmten Tag ausgewählt wurde. Zur Berechnung dieser Spalte werden die gesammelten Daten aus der Tabelle NATION_AUFGERUFEN verwendet.

```
STATS_NATION_AUFGERUFEN (
        DATUM                   DATE,
        NATION_ID               NUMBER,
        ANZAHL_AUFRUFE          NUMBER
);
```

Abbildung 89: Struktur der Tabelle STATS_NATION_AUFGERUFEN

Zur Berechnung der Statistik-Daten für die Tabelle STATS_NATION_AUFGERUFEN wird eine einfache SELECT-Anweisung im Statistiktool verwendet. Die gesammelten Daten werden nach den Spalten DATUM und NATION_ID gruppiert und die Berechnung der Spalten ANZAHL_AUFRUFE erfolgt durch die Funktion COUNT(). Diese SELECT-Anweisung berücksichtigt nur gesammelte Daten, deren Datum beispielsweise größer gleich 01.01.2013 ist.

```
SELECT TO_CHAR(datum, 'DD.MM.YYYY'),
            nation_id,
            COUNT(nation_id)
FROM nation_aufgerufen
WHERE datum >= TO_DATE('01.01.2013', 'DD.MM.YYYY')
GROUP BY TO_CHAR(datum, 'DD.MM.YYYY'), nation_id;
```

Abbildung 90: SELECT-Anweisung zur Tabelle STATS_NATION_AUFGERUFEN

DATUM	NATION_ID	ANZAHL_AUFRUFE
11.04.12	1	73
12.04.12	1	178
30.04.12	2	26
03.05.12	1	77
04.05.12	1	86
09.05.12	2	24
17.05.12	2	8
22.05.12	1	65
31.05.12	1	81
02.06.12	1	30
06.06.12	1	52
07.06.12	2	26
08.06.12	1	17
10.06.12	2	34
11.06.12	2	20

Abbildung 91: Teilinhalt der Tabelle STATS_NATION_AUFGERUFEN

(Stand: 01.08.2013, 19:00 Uhr)

4.2.1.1.7 Tabelle STATS_SQL_OPTIMIZER_FRAGEN

Die Statistik-Tabelle STATS_SQL_OPTIMIZER_FRAGEN enthält Statistik-Daten des SQL-Abfrageoptimierungstools. In dieser Tabelle werden nur Daten gespeichert, die in Bezug auf die von Benutzern ausgewählten Aufgaben bzw. SQL-Abfragen stehen. Beim SQL-Abfrageoptimierungstool ist es auch möglich, dass Benutzer eine SQL-Abfrage selbst eingeben und ausführen, anstatt eine SQL-Abfrage auszuwählen.

Die Spalten DATUM und AUFGABE_ID wurden als Primärschlüssel gesetzt. Der Wert der Spalte ANZAHL_AUFRUFE gibt an, wie oft eine Aufgabe von Benutzern im SQL-Abfrageoptimierungstool in einem bestimmten Tag ausgewählt wurde. Zur Berechnung dieser Spalte werden die gesammelten Daten aus der Tabelle SQL_OPTIMIZER_SCHEMA verwendet.

```
STATS_SQL_OPTIMIZER_FRAGEN (
        DATUM                   DATE,
        AUFGABE_ID              NUMBER,
        ANZAHL_AUFRUFE          NUMBER
);
```

Abbildung 92: Struktur der Tabelle STATS_SQL_OPTIMIZER_FRAGEN

Die Statistik-Daten für die Tabelle STATS_SQL_OPTIMIZER_FRAGEN werden durch eine einfache SELECT-Anweisung im Statistiktool berechnet, die in der folgenden Abbildung gezeigt wird. Die gesammelten Daten werden nach den Spalten DATUM und AUFGABE_ID gruppiert und die Berechnung der Spalten ANZAHL_AUFRUFE erfolgt durch die Funktion COUNT(). Diese SELECT-Anweisung berücksichtigt nur gesammelte Daten, deren Datum beispielsweise größer gleich 01.01.2013 ist.

```
SELECT TO_CHAR(datum, 'DD.MM.YYYY'),
            aufgabe_id,
            COUNT(aufgabe_id)
FROM sql_optimizer_schema
WHERE datum >= TO_DATE('01.01.2013', 'DD.MM.YYYY')
GROUP BY TO_CHAR(datum, 'DD.MM.YYYY'), aufgabe_id;
```

Abbildung 93: SELECT-Anweisung zur Tabelle STATS_SQL_OPTIMIZER_FRAGEN

DATUM	AUFGABE_ID	ANZAHL_AUFRUFE
20.01.13	-1	6
20.01.13	1	2
20.01.13	43	1
24.01.13	1	7
24.01.13	2	1
24.01.13	43	1
24.01.13	45	1
24.01.13	52	3
24.01.13	53	1
24.01.13	64	2
24.01.13	65	1
24.01.13	67	1
24.01.13	68	1
25.01.13	-1	1
25.01.13	1	2

Abbildung 94: Teilinhalt der Tabelle STATS_SQL_OPTIMIZER_FRAGEN

(Stand: 01.08.2013, 19:00 Uhr)

4.2.1.1.8 Tabelle STATS_SQL_OPTIMIZER_SCHEMA

Im Gegensatz zur Tabelle STATS_SQL_OPTIMIZER_FRAGEN enthält die Tabelle STATS_SQL_OPTIMIZER_SCHEMA Statistik-Daten nur über die von Benutzern ausgewählten Schemata im SQL-Abfrageoptimierungstool. Im SQL-Abfrageoptimierungs-tool stehen Aufgaben bzw. SQL-Abfragen von den Schemata „Fahrrad“ und „Fußball“ zur Auswahl.

Als Primärschlüssel wurden die Spalten DATUM und SCHEMA_ID festgelegt. Dazu gehören auch die Spalten ANZAHL_AUFRUFE, ANZAHL_SELECT_GESCHRIEBEN und ANZAHL_SELECT_AUSGEWAEHLT zur Tabelle. Die Werte dieser Spalten werden aus den gesammelten Daten der Tabelle SQL_OPTIMIZER_SCHEMA berechnet. Die Spalte ANZAHL_AUFRUFE gibt an, wie oft eine SQL-Abfrage aus einem bestimmten Schema an einem bestimmten Tag von Benutzern ausgeführt wurde. Die Spalten ANZAHL_SELECT_AUSGEWAEHLT zeigen, wie oft eine SQL-Abfrage von einem bestimmten Schema in einem bestimmten Tag im SQL-Abfrageoptimierungstool ausgewählt wurde.

Die Spalte ANZAHL_SELECT_GESCHRIEBEN zeigen dagegen, wie häufig die Benutzer SQL-Abfragen selbst eingegeben haben.

```
STATS_SQL_OPTIMIZER_SCHEMA (
        DATUM                                   DATE,
        SCHEMA_ID                               NUMBER,
        ANZAHL_AUFRUFE                          NUMBER,
        ANZAHL_SELECT_GESCHRIEBEN               NUMBER,
        ANZAHL_SELECT_AUSGEWAEHLT               NUMBER
);
```

Abbildung 95: Struktur der Tabelle STATS_SQL_OPTIMIZER_SCHEMA

Die Berechnung der Statistik-Daten für die Tabelle STATS_SQL_OPTIMIZER_SCHEMA erfolgt durch die folgende SELECT-Anweisung im Statistiktool. Die gesammelten Daten werden nach den Spalten DATUM und SCHEMA_ID gruppiert. In der Berechnung wird die Funktion COUNT() für die Spalte ANZAHL_AUFRUFE verwendet. Für die Spalten ANZAHL_SELECT_GESCHRIEBEN und ANZAHL_SELECT_AUSGEWAEHLT wird die Funktion SUM() in Verbindung mit dem CASE-Ausdruck verwendet. Bei der Berechnung wird der Wert aus der Spalte AUFGABE_ID der Tabelle SQL_OPTIMIZER_SCHEMA überprüft. Wenn in der Spalte der Wert -1 steht, bedeutet es, dass die SQL-Abfrage vom Benutzer selbst eingegeben wurde. Bei allen anderen Werten gilt es, dass die SQL-Abfrage ausgewählt wurde und diese Werte den SQL-Abfrage-IDs bzw. Aufgabe-IDs im SQL-Abfrageoptimierungstool entsprechen. Diese SELECT-Anweisung berücksichtigt nur gesammelte Daten, deren Datum beispielsweise größer gleich 01.01.2013 ist.

```
SELECT TO_CHAR(datum, 'DD.MM.YYYY'),
            schema_id,
            COUNT(schema_id),
            SUM(CASE WHEN aufgabe_id = -1 THEN 1 ELSE 0 END),
            SUM(CASE WHEN aufgabe_id != -1 THEN 1 ELSE 0 END)
FROM sql_optimizer_schema
WHERE datum >= TO_DATE('01.01.2013', 'DD.MM.YYYY')
GROUP BY TO_CHAR(datum, 'DD.MM.YYYY'), schema_id;
```

Abbildung 96: SELECT-Anweisung zur Tabelle STATS_SQL_OPTIMIZER_SCHEMA

DATUM	SCHEMA_ID	ANZAHL_AUFRUFE	ANZAHL_SELECT_GESCHRIEBEN	ANZAHL_SELECT_AUSGEWAEHLT
20.01.13	1	7	5	2
20.01.13	2	2	1	1
24.01.13	1	8	0	8
24.01.13	2	11	0	11
25.01.13	2	2	0	2
25.01.13	1	3	1	2
26.01.13	1	3	1	2
27.01.13	1	1	0	1
28.01.13	1	8	3	5
28.01.13	2	5	0	5
29.01.13	1	5	0	5
29.01.13	2	2	0	2
31.01.13	1	24	0	24
01.02.13	1	6	0	6
01.02.13	2	1	0	1

Abbildung 97: Teilinhalt der Tabelle STATS_SQL_OPTIMIZER_SCHEMA

(Stand: 01.08.2013, 19:00 Uhr)

4.2.1.1.9 Tabelle STATS_SQL_TRAINER_1_FRAGEN

Die Statistik-Tabelle STATS_SQL_TRAINER_1_FRAGEN enthält Statistik-Daten, die Informationen über die Aufrufe und die richtige Beantwortung von Fragen beim SQL-Trainer liefern.

Die Spalten DATUM und FRAGE_ID wurden als Primärschlüssel gesetzt. In die Spalte ANZAHL_AUFRUFE wird die Anzahl von Aufrufen einer bestimmten Frage an einem bestimmten Tag gespeichert. Die Spalten ANZAHL_RICHTIG_BEANTWORTET und ANZAHL_FALSCH_BEANTWORTET zeigen, wie oft eine Frage von Benutzern richtig oder falsch beantwortet wurde. Die Summe aus den Spalten ANZAHL_RICHTIG_BEANTWORTET und ANZAHL_FALSCH_BEANTWORTET entspricht dem Wert in der Spalte ANZAHL_AUFRUFE.

```
STATS_SQL_TRAINER_1_FRAGEN (
        DATUM                                   DATE,
        FRAGE_ID                                NUMBER,
        ANZAHL_AUFRUFE                          NUMBER,
        ANZAHL_RICHTIG_BEANTWORTET              NUMBER,
        ANZAHL_FALSCH_ BEANTWORTET              NUMBER
);
```

Abbildung 98: Struktur der Tabelle STATS_SQL_TRAINER_1_FRAGEN

Zur Berechnung der Statistik-Daten wird die folgende SELECT-Anweisung im Statistiktool verwendet, die auf die gesammelten Daten aus der Tabelle FRAGENAUSWERTUNG zugreift. Die gesammelten Daten werden nach den Spalten DATUM und FRAGE_ID gruppiert. Die Berechnung der Spalte ANZAHL_AUFRUFE wird mit der Funktion COUNT() durchgeführt. Die Spalten ANZAHL_RICHTIG_BEANTWORTET und ANZAHL_FALSCH_BEANTWORTET werden mit der Funktion SUM() in Verbindung mit dem CASE-Ausdruck berechnet. Bei der Berechnung wird der Wert aus der Spalte RICHTIG_BEANTWORTET der Tabelle FRAGENAUSWERTUNG ausgewertet. Diese Spalte kann nur den Wert 0 für falsch beantwortet oder 1 für richtig beantwortet annehmen.

Die SELECT-Anweisung enthält zwei Ausdrücke in der WHERE-Klausel. Der erste Ausdruck ist beispielsweise „DATUM >= TO_DATE('01.01.2013', 'DD.MM.YYYY')“ und bedeutet, dass die SELECT-Anweisung nur die gesammelten Daten berücksichtigt, deren Datum größer gleich 01.01.2013 ist. Der zweite Ausdruck „TOOL_ID = 4“ gibt an, von welchem Tool die Daten einbezogen werden sollen. In diesem Fall TOOL_ID ist immer gleich 4, weil die Nummer 4 die ID des SQL-Trainers in der Tabelle TOOLS ist. In der Tabelle FRAGENAUSWERTUNG sind auch die IDs von anderen Trainern gespeichert, wie MCT, SQL-Trainer2 und XQuery-Trainer.

```
SELECT TO_CHAR(datum, 'DD.MM.YYYY'),
            frage_id,
            COUNT(tool_id),
            SUM(CASE WHEN richtig_beantwortet = 1 THEN 1 ELSE 0 END),
            SUM(CASE WHEN richtig_beantwortet = 0 THEN 1 ELSE 0 END)
FROM fragenauswertung
WHERE datum >= TO_DATE('01.01.2013', 'DD.MM.YYYY') AND tool_id = 4
GROUP BY TO_CHAR(datum, 'DD.MM.YYYY'), frage_id;
```

Abbildung 99: SELECT-Anweisung zur Tabelle STATS_SQL_TRAINER_1_FRAGEN

DATUM	FRAGE_ID	ANZAHL_AUFRUFE	ANZAHL_RICHTIG_BEANTWORTET	ANZAHL_FALSCH_BEANTWORTET
03.04.12	88	1	0	1
03.04.12	16	1	0	1
03.04.12	222	1	0	1
03.04.12	196	1	0	1
03.04.12	221	2	0	2
03.04.12	194	1	0	1
04.04.12	70	1	0	1
04.04.12	10	1	0	1
04.04.12	246	1	0	1
05.04.12	245	1	0	1
05.04.12	20	1	0	1
06.04.12	50	2	0	2
06.04.12	38	2	0	2
06.04.12	44	1	0	1
07.04.12	66	1	0	1

Abbildung 100: Teilinhalt der Tabelle STATS_SQL_TRAINER_1_FRAGEN

(Stand: 01.08.2013, 19:00 Uhr)

4.2.1.1.10 Tabelle STATS_SQL_TRAINER_1_SCHEMA

Die Statistik-Tabelle STATS_SQL_TRAINER_1_SCHEMA enthält Statistik-Daten über die Auswahl von Schemata durch die Benutzer im SQL-Trainer. Die gesammelten Daten, die zur Berechnung der Statistik-Daten verwendet werden, stammen aus der Tabelle SQL_TRAINER_1_SCHEMA. Der Primärschlüssel der Tabelle bildet sich aus den Spalten DATUM und SCHEMA_ID. Der Wert der Spalte ANZAHL_AUFRUFE gibt an, wie oft ein Schema von Benutzern im SQL-Trainer an einem bestimmten Tag ausgewählt wurde.

```
STATS_SQL_TRAINER_1_SCHEMA (
        DATUM                   DATE,
        SCHEMA_ID               VARCHAR(30),
        ANZAHL_AUFRUFE          NUMBER
);
```

Abbildung 101: Struktur der Tabelle STATS_SQL_TRAINER_1_SCHEMA

Die Statistik-Daten für die Tabelle STATS_SQL_TRAINER_1_SCHEMA werden durch eine einfache SELECT-Anweisung im Statistiktool berechnet, die die folgende Abbildung zeigt. Die gesammelten Daten werden nach den Spalten DATUM und SCHEMA_ID gruppiert und die Berechnung der Spalten ANZAHL_AUFRUFE erfolgt durch die Funktion COUNT(). Diese SELECT-Anweisung berücksichtigt nur gesammelte Daten, deren Datum beispielsweise größer gleich 01.01.2013 ist.

```
SELECT TO_CHAR(datum, 'DD.MM.YYYY'),
            schema_id,
            COUNT(schema_id)
FROM sql_trainer_1_schema
WHERE datum >= TO_DATE('01.01.2013', 'DD.MM.YYYY')
GROUP BY TO_CHAR(datum, 'DD.MM.YYYY'), schema_id;
```

Abbildung 102: SELECT-Anweisung zur Tabelle STATS_SQL_TRAINER_1_SCHEMA

DATUM	SCHEMA_ID	ANZAHL_AUFRUFE
05.04.12	reisen	2
05.04.12	welt	1
05.04.12	busse	2
16.04.12	reisen	6
16.04.12	fahrrad	40
18.04.12	reisen	16
18.04.12	fahrrad	27
19.04.12	busse	13
19.04.12	theater	3
20.04.12	busse	20
22.04.12	fahrrad	2
23.04.12	welt	3
24.04.12	reisen	2
28.04.12	buses	10
30.04.12	busse	12

Abbildung 103: Teilinhalt der Tabelle STATS_SQL_TRAINER_1_SCHEMA

(Stand: 01.08.2013, 19:00 Uhr)

4.2.1.1.11 Tabelle STATS_SQL_TRAINER_FRAGEN

Die Statistik-Tabelle STATS_SQL_TRAINER_FRAGEN enthält Statistik-Daten, die Informationen über die Aufrufe und die richtige Beantwortung von Fragen beim SQL-Trainer 2 angeben.

Zur Eindeutigkeit der Datensätze wurden die Spalten DATUM und FRAGE_ID als Primärschlüssel gesetzt. In die Spalte ANZAHL_AUFRUFE wird die Anzahl von Aufrufen einer bestimmten Frage eines bestimmten Tages gespeichert, die der Summe der Spalten ANZAHL_RICHTIG_BEANTWORTET und ANZAHL_FALSCH_BEANTWORTET entspricht. Durch die Spalten ANZAHL_RICHTIG_BEANTWORTET und ANZAHL_FALSCH_BEANTWORTET wird angezeigt, wie oft eine Frage von Benutzern richtig oder falsch beantwortet wurde.

```
STATS_SQL_TRAINER_FRAGEN (
        DATUM                                   DATE,
        FRAGE_ID                                NUMBER,
        ANZAHL_AUFRUFE                          NUMBER,
        ANZAHL_RICHTIG_BEANTWORTET              NUMBER,
        ANZAHL_FALSCH_ BEANTWORTET              NUMBER
);
```

Abbildung 104: Struktur der Tabelle STATS_SQL_TRAINER_FRAGEN

Die Berechnung der Statistik-Daten erfolgt durch die folgende SELECT-Anweisung im Statistiktool, die auf die gesammelten Daten aus der Tabelle FRAGENAUSWERTUNG zugreift. Die gesammelten Daten werden nach den Spalten DATUM und FRAGE_ID gruppiert. Die Spalte ANZAHL_AUFRUFE wird mit der Funktion COUNT() berechnet. Bei der Berechnung der Spalten ANZAHL_RICHTIG_BEANTWORTET und ANZAHL_FALSCH_BEANTWORTET wird die Funktion SUM() in Verbindung mit dem CASE-Ausdruck verwendet. Bei der Berechnung wird der Wert aus der Spalte RICHTIG_BEANTWORTET der Tabelle FRAGENAUSWERTUNG ausgewertet. Diese Spalte kann nur den Wert 0 für „falsch beantwortet“ oder den Wert 1 für „richtig beantwortet“ annehmen.

Die SELECT-Anweisung besitzt zwei Ausdrücke in der WHERE-Klausel. Der erste Ausdruck lautet beispielsweise „DATUM >= TO_DATE('01.01.2013', 'DD.MM.YYYY')“ und bedeutet, dass die SELECT-Anweisung nur die gesammelten Daten berücksichtigt, deren Datum größer gleich 01.01.2013 ist. Der zweite Ausdruck „TOOL_ID = 5“ gibt an, von welchem Tool die Daten einbezogen werden sollen. In diesem Fall TOOL_ID ist es immer gleich 5, weil die Nummer 5 die ID des SQL-Trainers 2 in der Tabelle TOOLS ist. In der Tabelle FRAGENAUSWERTUNG sind auch die IDs von anderen Trainern gespeichert, wie MCT, SQL-Trainer und XQuery-Trainer.

```
SELECT TO_CHAR(datum, 'DD.MM.YYYY'),
            frage_id,
            COUNT(tool_id),
            SUM(CASE WHEN richtig_beantwortet = 1 THEN 1 ELSE 0 END),
            SUM(CASE WHEN richtig_beantwortet = 0 THEN 1 ELSE 0 END)
FROM fragenauswertung
WHERE datum >= TO_DATE('01.01.2013', 'DD.MM.YYYY') AND tool_id = 5
GROUP BY TO_CHAR(datum, 'DD.MM.YYYY'), frage_id;
```

Abbildung 105: SELECT-Anweisung zur Tabelle STATS_SQL_TRAINER_FRAGEN

DATUM	FRAGE_ID	ANZAHL_AUFRUFE	ANZAHL_RICHTIG_BEANTWORTET	ANZAHL_FALSCH_BEANTWORTET
15.04.12	946	2	1	1
15.04.12	864	1	1	0
15.04.12	932	1	0	1
08.05.12	987	7	0	7
08.05.12	923	3	2	1
08.05.12	843	2	1	1
08.05.12	983	3	0	3
09.05.12	991	2	1	1
09.05.12	952	5	0	5
09.05.12	988	1	0	1
09.05.12	987	3	0	3
12.05.12	1075	4	0	4
14.05.12	832	1	0	1
18.05.12	835	1	0	1
18.05.12	930	2	1	1

Abbildung 106: Teilinhalt der Tabelle STATS_SQL_TRAINER_FRAGEN

(Stand: 01.08.2013, 19:00 Uhr)

4.2.1.1.12 Tabelle STATS_SQL_TRAINER_SCHEMA

In der Statistik-Tabelle STATS_SQL_TRAINER_SCHEMA werden Statistik-Daten gespeichert, die Informationen über die Auswahl von Schemata durch die Benutzer im SQL-Trainer 2 angeben. Die gesammelten Daten, die zur Berechnung der Statistik-Daten verwendet werden, stammen aus der Tabelle SQL_TRAINER_SCHEMA. Die Spalten DATUM und SCHEMA_ID wurden als Primärschlüssel gesetzt. Der Wert der Spalte AN-

ZAHL_AUFRUFE zeigt, wie oft ein Schema von Benutzern im SQL-Trainer 2 an einem bestimmten Tag ausgewählt wurde.

```
STATS_SQL_TRAINER_SCHEMA (
        DATUM                   DATE,
        SCHEMA_ID               NUMBER,
        ANZAHL_AUFRUFE          NUMBER
);
```

Abbildung 107: Struktur der Tabelle STATS_SQL_TRAINER_SCHEMA

Die folgende SELECT-Anweisung berechnet die Statistik-Daten für die Tabelle STATS_SQL_TRAINER_SCHEMA im Statistiktool. Die gesammelten Daten werden nach den Spalten DATUM und SCHEMA_ID gruppiert und die Berechnung der Spalten ANZAHL_AUFRUFE erfolgt durch die Funktion COUNT(). Diese SELECT-Anweisung berücksichtigt nur gesammelte Daten, deren Datum beispielsweise größer gleich 01.01.2013 ist.

```
SELECT TO_CHAR(datum, 'DD.MM.YYYY'),
            schema_id,
            COUNT(schema_id)
FROM sql_trainer_schema
WHERE datum >= TO_DATE('01.01.2013', 'DD.MM.YYYY')
GROUP BY TO_CHAR(datum, 'DD.MM.YYYY'), schema_id;
```

Abbildung 108: SELECT-Anweisung zur Tabelle STATS_SQL_TRAINER_SCHEMA

DATUM	SCHEMA_ID	ANZAHL_AUFRUFE
04.04.12	1	1
14.04.12	1	2
19.04.12	8	10
23.04.12	8	1
29.04.12	1	9
30.04.12	1	2
11.05.12	1	8
12.05.12	6	65
15.05.12	1	22
17.05.12	1	2
20.05.12	2	1
28.05.12	1	2
05.06.12	8	3
05.06.12	6	2
13.06.12	1	16

Abbildung 109: Teilinhalt der Tabelle STATS_SQL_TRAINER_SCHEMA

(Stand: 01.08.2013, 19:00 Uhr)

4.2.1.1.13 Tabelle STATS_XQUERY_FRAGEN

Die Statistik-Tabelle STATS_XQUERY_FRAGEN enthält Statistik-Daten, die Informationen über die Aufrufe und die richtige Beantwortung von Fragen beim XQuery-Trainer des EDB-Portals angeben.

Die Spalten DATUM und FRAGE_ID wurden als Primärschlüssel gesetzt. In die Spalte ANZAHL_AUFRUFE wird die Anzahl von Aufrufen einer bestimmten Frage in einem bestimmten Tag des XQuery-Trainers gespeichert, die der Summe der Spalten ANZAHL_RICHTIG_BEANTWORTET und ANZAHL_FALSCH_BEANTWORTET entspricht. Durch die Spalten ANZAHL_RICHTIG_BEANTWORTET und ANZAHL_FALSCH_BEANTWORTET wird zeigen, wie oft eine Frage von Benutzern richtig oder falsch beantwortet wurde.

```
STATS_XQUERY_FRAGEN (
        DATUM                                   DATE,
        FRAGE_ID                                NUMBER,
        ANZAHL_AUFRUFE                          NUMBER,
        ANZAHL_RICHTIG_BEANTWORTET              NUMBER,
        ANZAHL_FALSCH_ BEANTWORTET              NUMBER
);
```

Abbildung 110: Struktur der Tabelle STATS_XQUERY_FRAGEN

Die Statistik-Daten werden durch die folgende SELECT-Anweisung im Statistiktool berechnet. Die SELECT-Anweisung greift auf die gesammelten Daten aus der Tabelle FRAGENAUSWERTUNG zu. Die gesammelten Daten werden nach den Spalten DATUM und FRAGE_ID gruppiert und die Berechnung der Spalte ANZAHL_AUFRUFE erfolgt mit der Funktion COUNT(). Die Spalten ANZAHL_RICHTIG_BEANTWORTET und ANZAHL_FALSCH_BEANTWORTET werden durch die Funktion SUM() in Verbindung mit dem CASE-Ausdruck berechnet. Bei der Berechnung wird der Wert aus der Spalte RICHTIG_BEANTWORTET der Tabelle FRAGENAUSWERTUNG ausgewertet. Diese Spalte kann nur die Werte 0 für falsch beantwortet oder 1 für richtig beantwortet annehmen.
Die SELECT-Anweisung beinhaltet zwei Ausdrücke in der WHERE-Klausel. Der erste Ausdruck lautet beispielsweise „DATUM >= TO_DATE('01.01.2013', 'DD.MM.YYYY')“ und bedeutet, dass die SELECT-Anweisung nur die gesammelten Daten berücksichtigt, deren Datum größer gleich 01.01.2013 ist. Der zweite Ausdruck „TOOL_ID = 13“ gibt an, von welchem Tool die Daten einbezogen werden sollen. In diesem Fall TOOL_ID ist es immer gleich 13, weil die Nummer 13 die ID des XQuery-Trainers in der Tabelle TOOLS ist. In der Tabelle FRAGENAUSWERTUNG sind auch die IDs von anderen Trainern gespeichert, wie MCT, SQL-Trainer und SQL-Trainer 2.

```
SELECT TO_CHAR(datum, 'DD.MM.YYYY'),
            frage_id,
            COUNT(tool_id),
            SUM(CASE WHEN richtig_beantwortet = 1 THEN 1 ELSE 0 END),
            SUM(CASE WHEN richtig_beantwortet = 0 THEN 1 ELSE 0 END)
FROM fragenauswertung
WHERE datum >= TO_DATE('01.01.2013', 'DD.MM.YYYY') AND tool_id = 13
GROUP BY TO_CHAR(datum, 'DD.MM.YYYY'), frage_id;
```

Abbildung 111: SELECT-Anweisung zur Tabelle STATS_XQUERY_FRAGEN

DATUM	FRAGE_ID	ANZAHL_AUFRUFE	ANZAHL_RICHTIG_BEANTWORTET	ANZAHL_FALSCH_BEANTWORTET
12.05.12	53	1	0	1
12.05.12	99	2	1	1
19.05.12	56	3	0	3
26.06.12	7	2	0	2
26.06.12	13	1	0	1
26.06.12	14	1	0	1
26.06.12	15	1	0	1
26.06.12	16	1	0	1
26.06.12	18	1	0	1
26.06.12	19	1	0	1
26.06.12	20	1	0	1
26.06.12	54	7	1	6
26.06.12	99	2	0	2
08.07.12	56	2	0	2
08.07.12	57	2	0	2

Abbildung 112: Teilinhalt der Tabelle STATS_XQUERY_FRAGEN

(Stand: 01.08.2013, 19:00 Uhr)

4.2.1.1.14 Tabelle STATS_XQUERY_SCHEMA

In die Statistik-Tabelle STATS_XQUERY_SCHEMA werden Statistik-Daten über die Auswahl von Schemata durch die Benutzer im XQuery-Trainer gespeichert. Die gesammelten Daten, die zur Berechnung der Statistik-Daten verwendet werden, stammen aus der Tabelle XQUERY_SCHEMA. Der Primärschlüssel der Tabelle bildet sich aus den Spalten DATUM und SCHEMA_ID. Der Wert der Spalte ANZAHL_AUFRUFE gibt an, wie oft ein Schema von Benutzern im XQuery-Trainer an einem bestimmten Tag ausgewählt wurde.

```
STATS_XQUERY_SCHEMA (
        DATUM                   DATE,
        SCHEMA_ID               NUMBER,
        ANZAHL_AUFRUFE          NUMBER
);
```

Abbildung 113: Struktur der Tabelle STATS_XQUERY_SCHEMA

Die Statistik-Daten für die Tabelle STATS_XQUERY_SCHEMA werden durch eine einfache SELECT-Anweisung im Statistiktool berechnet, die die folgende Abbildung zeigt. Die gesammelten Daten werden nach den Spalten DATUM und SCHEMA_ID gruppiert und die Berechnung der Spalten ANZAHL_AUFRUFE erfolgt durch die Funktion COUNT(). Diese SELECT-Anweisung berücksichtigt nur gesammelte Daten, deren Datum beispielsweise größer gleich 01.01.2013 ist.

```
SELECT TO_CHAR(datum, 'DD.MM.YYYY'),
            schema_id,
            COUNT(schema_id)
FROM xquery_schema
WHERE datum >= TO_DATE('01.01.2013', 'DD.MM.YYYY')
GROUP BY TO_CHAR(datum, 'DD.MM.YYYY'), schema_id;
```

Abbildung 114: SELECT-Anweisung zur Tabelle STATS_XQUERY_SCHEMA

DATUM	SCHEMA_ID	ANZAHL_AUFRUFE
05.05.12	1	1
12.05.12	1	8
19.05.12	1	4
03.06.12	1	3
26.06.12	1	21
26.06.12	2	25
02.07.12	1	1
08.07.12	1	7
22.08.12	1	1
29.08.12	1	2
20.09.12	1	3
23.09.12	1	13
24.09.12	1	3
02.10.12	1	1
29.10.12	1	1

Abbildung 115: Teilinhalt der Tabelle STATS_XQUERY_SCHEMA

(Stand: 01.08.2013, 19:00 Uhr)

4.2.1.2 Überprüfung der Statistik-Daten

Nach der Erstellung der SELECT-Anweisungen wurde nach einer Methode gesucht, um die Richtigkeit der berechneten Statistik-Daten in allen Statistik-Tabellen zu überprüfen. Die einfachste Methode ergab sich aus dem Zählen der Datensätze. Man kann die Statistik-Daten überprüfen, indem die Datensätze durch eine einfache SELECT-Anweisung mit der Funktion COUNT() gezählt werden.

Zuerst werden nur fünf Statistik-Daten berechnet und als Testdaten in die Statistik-Tabellen eingespeichert, um die Überprüfung übersichtlicher zu gestalten. Alle fünf Datensätze werden dann auf ihre Richtigkeit überprüft.

Als Beispiel wird die Statistik-Tabelle STATS_NATION_AUFGERUFEN genommen. Die folgende SELECT-Anweisung gibt den Inhalt der Tabelle mit den fünf Testdaten aus.

```
SELECT *
FROM STATS_NATION_AUFGERUFEN
ORDER BY DATUM;
```

Abbildung 116: SELECT-Anweisung zur Ausgabe der Testdaten

	DATUM	NATION_ID	ANZAHL_AUFRUFE
1	11.04.12	1	73
2	12.04.12	1	178
3	30.04.12	2	26
4	03.05.12	1	77
5	04.05.12	1	86

Abbildung 117: Inhalt der Tabelle STATS_NATION_AUFGERUFEN mit den Testdaten

Aus der Tabelle STATS_NATION_AUFGERUFEN wird zu diesem Beispiel nur der Inhalt des dritten Datensatzes kontrolliert. Die Berechnung der Statistik-Daten von der dritten Zeile ergab, dass die Nation mit der ID 2 am 30.04.2012 insgesamt 26 Male aufgerufen wurde. Um die Richtigkeit dieser Informationen zu überprüfen, muss die Tabelle NATION_AUFGERUFEN abgefragt werden, die die gesammelten Daten zu der Statistik-Tabelle STATS_NATION_AUFGERUFEN enthält und zur Berechnung der Statistik-Daten verwendet wurde. In die Abfrage der Tabelle NATION_AUFGERUFEN müssen die Spalten NATION_ID und DATUM in die WHERE-Klausel der SELECT-Anweisung eingesetzt werden. Die Abfrage kann dann mit den folgenden einfachen SELECT-Anweisungen erfolgen, in der die Funktion COUNT() verwendet wird.

```
SELECT COUNT(*)
FROM NATION_AUFGERUFEN
WHERE TO_CHAR(DATUM, 'DD.MM.YYYY') = '30.04.2012' AND NATION_ID = 2;
```

Abbildung 118: SELECT-Anweisung zur Überprüfung der Statistik-Daten

COUNT(*)
26

Abbildung 119: Ergebnis der SELECT-Anweisung

Das Ergebnis der SELECT-Anweisungen entspricht dem berechneten Wert aus der Spalte ANZAHL_AUFRUFE von der Tabelle STATS_NATION_AUFGERUFEN. Damit kommt man zum gleichen Ergebnis, dass die Nation mit der ID 2 am 30.04.2012 insgesamt 26 Male aufgerufen wurde und dass die Berechnung der Statistik-Daten richtig durchgeführt wurde.

4.2.2 Anwendungsschicht

4.2.2.1 Servlets

Zur Durchführung von Anwendungslogik-Aufgaben wurden sieben Servlets im Statistiktool implementiert. Die Java-Klassen, aus denen die Servlets erzeugt werden, befinden sich im Paket stats und heißen ServletStatsAktualisieren.java, ServletStatsAlleTools.java, ServletStatsMCT.java, ServletStatsSQLOPT.java, ServletStatsSQLTrainer.java, ServletStatsSQLTrainer2.java und ServletStatsXQueryTrainer.java.

Bei den Anfragen von JSP werden entsprechende Parameter an die Servlets übergeben, die über HTTP-POST-Anfragen übertragen werden. Somit werden die Parameter in der Methode doPost der Servlets ausgewertet und verarbeitet. Die Parameter werden vom Anfrageobjekt (Request) ausgelesen.

Die Anwendungslogik-Aufgaben bestehen darin, zuerst zu überprüfen, ob die von JSP gesendeten Parameter gültige Daten enthalten. Danach werden die angeforderten Statistik-Daten aus den Statistik-Tabellen gelesen und in einem JSON-Objekt gespeichert. Zur Erzeugung eines JSON-Objekts ist es notwendig, das Paket org.json zu importieren. Anschließend wird das JSON-Objekt in das Antwortobjekt (Response) geschrieben und an JSP gesendet. Bei dem Auslösen einer Exception wird diese in Try-Catch-Anweisungen abgefangen. Die erzeugte Fehlermeldung wird in das Element „fehlermeldung“ vom JSON-Objekt gespeichert.

In den Servlets werden immer zwei SELECT-Anweisungen ausgeführt. Die Ergebnismenge der ersten SELECT-Anweisung enthält alle angeforderten Statistik-Daten. Die Ergebnismenge der zweiten SELECT-Anweisung besteht aus der Summe der angeforderten Statistik-

Daten. Die einzige Ausnahme ist das Servlet ServletStatsAktualisieren, das für die Aktualisierung der Statistik-Tabellen zuständig ist und keine Statistik-Daten selektieren muss.

Die Servlets des Statistiktools erzeugen ein JSON-Objekt mit dem folgenden Quellcode in der Abbilddung 120. Für dieses Beispiel wurde das Servlet ServletStatsSQLOPT verwendet. Das Element „fehlermeldung“ im JSON-Objekt wird nur mit einem Wert belegt, falls ein Fehler im Servlet aufgetreten ist. Das Element „trainer“ zeigt, aus welchem Trainer die Statistik-Daten stammen. Bei dem Element „abfrage“ wird die vom Benutzer ausgewählten Abfrage bzw. Option auf der Benutzeroberfläche des Statistiktools gespeichert. In diesem Beispiel lautet die Option „Schema mit der größten Anzahl von Aufrufen“. Die Elemente „von“ und „bis“ sind den Zeitraum der Abfrage, der vom Benutzer bestimmt wird. In diesem Bespiel liegt der Zeitraum der Abfrage zwischen 01.03.2013 und 07.03.2013 (von=01.03.2013 und bis=07.03.2013). Der Array „datenArray“ enthält die angeforderten Statistik-Daten, während in dem Array „summeArray“ die Summe der angeforderten Statistik-Daten gespeichert ist. Das JSON-Objekt des Servlets ServletStatsAktualisieren enthält dagegen nur das Element „fehlermeldung“, weil es keine Daten an JSP senden muss.

```
res = st.executeQuery(query);
res1 = st1.executeQuery(result);

// AUSGABE
JSONObject json  = new JSONObject();
JSONArray datenArray = new JSONArray();
JSONArray summeArray = new JSONArray();
JSONObject daten;
JSONObject summeDaten;

try {
        // Zusaetzliche Informationen
        json.put("fehlermeldung", "");          // Keine Fehlermeldung
        json.put("trainer", tool);              // ausgewählter Trainer
        json.put("abfrage", abfrage);           // ausgewählte Abfrage
        json.put("von", von);                   // Datum (von)
        json.put("bis", bis);                   // Datum (bis)

        // Daten in datenArray
        while(res.next()) {
                daten = new JSONObject();
                daten.put("datum", res.getString(1));
                daten.put("id", res.getInt(2));
                daten.put("inhalt", res.getString(3));
                daten.put("anzahl", res.getInt(4));
                datenArray.put(daten);
        }
        // Zusammenfassung in summeArray
        while(res1.next()) {
                summeDaten = new JSONObject();
                summeDaten.put("id", res1.getInt(1));
                summeDaten.put("inhalt", res1.getString(2));
                summeDaten.put("summe", res1.getInt(3));
                summeArray.put(summeDaten);
        }
        json.put("datenArray", datenArray);     // datenArray in JSON-Objekt speichern.
        json.put("summeArray", summeArray); // summeArray in JSON-Objekt speichern.

}
catch (JSONException jse) {
        json.put("fehlermeldung", "JSONException" + jse.getMessage());
}
st.close();
st1.close();
con.close();
response.setContentType("application/json");
response.setCharacterEncoding("UTF-8");
response.getWriter().write(json.toString());
```

Abbildung 120: Quellcode zur Erzeugung eines JSON-Objekts

Aus dem Quellcode wird ein JSON-Objekt mit der Struktur erzeugt, die in der folgenden Abbildung dargestellt wird.


```
{
   "fehlermeldung": "",
   "trainer": "sqlopt",
   "abfrage": 1,
   "von": "1.3.2013",
   "bis": "7.3.2013",
   "datenArray": [
      {
         "id": 1,
         "anzahl": 7,
         "inhalt": "FAHRRAD",
         "datum": "02.03.2013"
      },
      {
         "id": 2,
         "anzahl": 1,
         "inhalt": "FUSSBALL",
         "datum": "04.03.2013"
      },
      {
         "id": 2,
         "anzahl": 9,
         "inhalt": "FUSSBALL",
         "datum": "07.03.2013"
      },
      {
         "id": 1,
         "anzahl": 1,
         "inhalt": "FAHRRAD",
         "datum": "07.03.2013"
      }
   ],
   "summeArray": [
      {
         "id": 2,
         "inhalt": "FUSSBALL",
         "summe": 10
      },
      {
         "id": 1,
         "inhalt": "FAHRRAD",
         "summe": 8
      }
   ]
}
```


Abbildung 121: Struktur des JSON-Objekts

Alle Servlets müssen auf das Statistiktool-Schema zugreifen, um die benötigten Daten zu holen. Aus diesem Grund wurden bei allen Servlets Initialisierungsparameter im Deployment Descriptor gesetzt, in denen der URL (Uniform Resource Locator), der Benutzername und das Passwort der Datenbank des Statistiktool-Schemas gespeichert sind, damit die Servlets eine Verbindung mit dieser Datenbank herstellen können. Auf den Deployment Descriptor kann von außerhalb des Web-Servers nicht zugegriffen werden, sodass keine Sicherheitsrisiken entstehen können.

4.2.2.1.1 ServletStatsAktualisieren

Im Servlet ServletStatsAktualisieren werden die Aktualisierung der Statistik-Tabellen und die Berechnung der Statistik-Daten implementiert. Dieses Servlet enthält mehrere Methoden, die systematisch nacheinander aufgerufen werden. Jede Methode erstellt und speichert die Statistik-Daten für eine bestimmte Statistik-Tabelle.

Das Servlet ServletStatsAktualisieren erhält zuerst aus der JSP-Anfrage nur den Parameter „bis“, der das Datum der letzten durchgeführten Statistik-Berechnung enthält. Somit ist es bekannt, bis wann die Statistik-Daten berechnet wurden und diese in den Statistik-Tabellen vorhanden sind. Der Wert des Parameters wird in die Variable „bis“ gespeichert. Bei der Aktualisierung der Statistik-Tabellen sollen dann nur die gesammelten Daten in die Berechnung einbezogen werden, deren Datum größer gleich als das Datum in der Variable „bis“ ist. Falls der Wert der Variable 0 ist, bedeutet das, dass noch keine Statistik-Daten berechnet wurden und die Statistik-Tabellen leer sind. Somit müssen alle gesammelten Daten in der Berechnung der Statistik-Daten berücksichtigt werden. Danach wird eine Verbindung mit der Datenbank hergestellt, indem ein Connection-Objekt erzeugt wird und bei dem Aufruf der Methode als Parameter übergeben wird.

Im Folgenden werden alle Methoden des Servlets in der Reihenfolge ihres Aufrufs beschrieben.

Methode: letzterTagLoeschen()

Die Methode letzterTagLoeschen() wird nur aufgerufen, wenn der Wert von „bis“ ungleich 0 ist. Diese Methode löscht alle Statistik-Daten aller Statistik-Tabellen, deren Datum mit dem Wert der Variable „bis“ übereinstimmt. Der Grund dafür ist, dass bei der letzten Aktualisierung der Statistik-Daten nicht alle gesammelten Daten von diesem Tag in der Berechnung berücksichtigt werden können, weil Daten an diesem Tag noch gesammelt werden. Ein Beispiel dafür wäre, wenn die Aktualisierung am 01.08.2013 um 13:00 Uhr ausgeführt wird, können nur die gesammelten Daten bis 13:00 Uhr in die Berechnung einbezogen werden, weil die Daten vom 01.08.2013 zwischen 13:01 Uhr und 23:59 Uhr noch fehlen. Die Daten mit dem Datum, die gelöscht werden, werden bei der Aktualisierung der Statistik-Daten in den anderen Methoden neu berechnet.

Methode: toolAufrufeBerechnen()

Die Methode toolAufrufeBerechnen() berechnet die Anzahl von Aufrufen und Beenden aller Trainer bzw. Tools auf dem EDB-Portal und speichert diese in die Tabelle STATS_TOOL_AUFGERUFEN. Die Variable „bis“ mit dem Connection-Objekt wird der Methode als Parameter übergeben. Mit diesem Wert wird die WHERE-Klausel der SELECT-Anweisung[11], die die Statistik-Daten berechnet, dynamisch erzeugt. Nach der Ausführung der SELECT-Anweisung wird die Ergebnismenge durch INSERT-Anweisungen in die STATS_TOOL_AUFGERUFEN gespeichert. Die folgende Abbildung zeigt die Methode toolAufrufeBerechnen().

[11] *Abbildung 78, S. 116.*

```
public void toolAufrufeBerechnen(Connection con, String bis) {
        Statement st;
        PreparedStatement insert;
        ResultSet res;
        String whereKlausel;

        if(bis.equals("0")){
                // Keine WHERE-Klausel nötig, da alle gesammelten Daten berücksichtigt werden müssen.
                whereKlausel = "";
        }
        else {
                whereKlausel = "WHERE ta.datum >= TO_DATE('"+ bis + "', 'DD.MM.YYYY') ";
        }

        try{
                st = con.createStatement();
                res = st.executeQuery("SELECT TO_CHAR(ta.datum, 'DD.MM.YYYY'), " +
                                        "ta.tool_id, " +
                                        "COUNT(ta.tool_id), "+
                                        "(SELECT COUNT(tb.tool_id) " +
                                        "FROM tool_beendet tb "+
                                        "WHERE TO_CHAR(ta.datum, 'DD.MM.YYYY') =  " +
                                        "TO_CHAR(tb.datum, 'DD.MM.YYYY') " +
                                        "AND ta.tool_id = tb.tool_id) " +
                                    "FROM tool_aufgerufen ta " +
                                     whereKlausel + " " +
                                    "GROUP BY TO_CHAR(ta.datum, 'DD.MM.YYYY'), ta.tool_id");

                insert = con.prepareStatement("INSERT INTO STATS_TOOL_AUFGERUFEN " +
                                            " (datum, tool_id, anzahl_aufrufe, anzahl_beendet) " +
                                            "VALUES (TO_DATE(?, 'DD.MM.YYYY'), ?, ?, ?)");

                while(res.next()) {
                        insert.setString(1, res.getString(1));  // datum
                        insert.setInt(2, res.getInt(2));          // tool_id
                        insert.setInt(3, res.getInt(3));         // anzahl_aufrufe
                        insert.setInt(4, res.getInt(4));         // anzahl_beendet
                        insert.executeUpdate();
                }
                st.close();
                insert.close();

        }catch(SQLException e){
                System.out.println("SQLException bei der Methode toolAufrufeBerechnen(): "+
                                e.getMessage());
        }
}
```

Abbildung 122: Methode toolAufrufeBerechnen()

Methode: anzahlAufrufeBerechnen()

Die Methode anzahlAufrufeBerechnen() wird zur Berechnung der Statistik-Daten von verschiedenen Statistik-Tabellen verwendet.

Diese Methode speichert die Statistik-Daten in die Statistik-Tabellen:

- STATS_NATION_AUFRUFEN
- STATS_SQL_TRAINER_SCHEMA
- STATS_SQL_TRAINER_1_SCHEMA
- STATS_XQUERY_SCHEMA
- STATS_SQL_OPTIMIZER_FRAGEN
- STATS_MCT_UNI
- STATS_MCT_KATEGORIEN

Diese Tabellen haben die gleiche Struktur, so dass die Berechnung der Statistik-Daten in einer einzigen Methode implementiert werden kann. Dafür müssen aber die entsprechenden Tabellen mit den Spalten und die Statistik-Tabelle als Parameter übergeben werden, die in der SELECT-Anweisung[12] der Methode verwendet werden. Dazu wird auch das Connection-Objekt mit der Variable „bis“ übergeben.

In dieser Methode werden auch die WHERE-Klausel und die SELECT-Anweisung dynamisch erzeugt. Der wichtige Aspekt dieser Methode liegt in der While-Schleife, weil zwischen den Tabellen unterschieden werden muss. Der Grund dafür ist, dass die Spalte SCHEMA_ID einiger Tabellen aus verschiedenen Datentypen besteht. In der Tabelle SQL_TRAINER_1_SCHEMA ist diese Spalte vom Datentyp String und dort steht nicht die ID des Schemas selbst, sondern der Name des Schemas. In der Tabelle XQUERY_SCHEMA ist die Spalte SCHEMA_ID auch vom Datentyp String, der aber in den Datentyp Integer umgewandelt werden kann. Bei alle anderen Tabellen, wie die Spalte SCHEMA_ID, ist die Spalte vom Datentyp Integer und es muss keine Konvertierung durchgeführt werden.

[12] *Abb. 84, S. 120; Abb. 87, S. 122; Abb. 90, S. 123; Abb. 93, S. 125; Abb. 102, S. 130; Abb. 108, S. 134; Abb. 114, S. 138.*

Die folgende Abbildung zeigt die While-Schleife der Methode anzahlAufrufeBerechnen().

```
while(res.next()) {
        insert.setString(1, res.getString(1));  // datum
        // In der Tabelle SQL_TRAINER_1_SCHEMA ist die Spalte Schema_ID vom Datentyp String und
steht der Name des Schemas nicht die ID selbst.
        if(tabelle.equals(tabSQLT1Schema)){
                insert.setString(2, res.getString(2));
        }
        // In der Tabelle XQUERY_SCHEMA ist die Spalte Schema_ID vom Datentyp String, ID in String
wird in Integer konvertiert.
        else if(tabelle.equals(tabXQuerySchema)){
                try {
                        insert.setInt(2, Integer.parseInt(res.getString(2)));
                } catch(NumberFormatException nfe){
                        System.out.println("ParseException bei der Datei StatsAktualisieren.jsp: "+
nfe.getMessage());
                }
        }
        else {
        // Fuer alle anderen Tabellen.
                insert.setInt(2, res.getInt(2));
        }
                insert.setInt(3, res.getInt(3));   // anzahl_aufrufe
                insert.executeUpdate();
}
```

Abbildung 123: While-Schleife in der Methode anzahlAufrufeBerechnen()

Methode: anzahlFragenBerechnen()

Die Methode anzahlFragenBerechnen() berechnet, wie oft Fragen von den Tabellen SQL_TRAINER_SCHEMA, SQL_TRAINER_1_SCHEMA und XQUERY_SCHEMA aufgerufen wurden.
Die Statistik-Daten werden in die folgenden Statistik-Tabellen gespeichert:

- STATS_SQL_TRAINER_FRAGEN
- STATS_SQL_TRAINER_1_FRAGEN
- STATS_XQUERY_FRAGEN

Die Methode hat als Parameter das Connection-Objekt, die Variable „bis“ die Statistik-Tabellen, in die die Statistik-Daten gespeichert werden, und auch die Tabelle, die die gesam-

melten Daten zur Berechnung enthält. Wie bei den anderen Methoden werden in der Methode anzahlFragenBerechnen() die WHERE-Klausel und die SELECT-Anweisung[13] dynamisch erzeugt.

Methode: anzahlSQLOptSchemaBerechnen()

Die Methode anzahlSQLOptSchemaBerechnen() berechnet, wie oft ein Schema im SQL-Abfrageoptimierungstool aufgerufen wurde. Die berechneten Statistik-Daten werden in die Tabelle STATS_SQL_OPTIMIZER_SCHEMA gespeichert.

Die Berechnung von Schema-Aufrufe konnte beim SQL-Abfrageoptimierungstool nicht von der Methode anzahlAufrufeBerechnen() durchgeführt werden. Die Statistik-Tabelle STATS_SQL_OPTIMIZER_SCHEMA besitzt eine andere Struktur im Vergleich mit den anderen Statistik-Tabellen, in denen die Anzahl der Aufrufe von Schema auch gespeichert werden. Der Unterschied liegt daran, dass im SQL-Abfrageoptimierungstool SQL-Abfragen von einem Schema ausgewählt oder selbst eingegeben werden können.

Wie bei den anderen Methoden, werden das Connection-Objekt und die Variable „bis“ als Parameter übergeben und die WHERE-Klausel sowie die SELECT-Anweisung[14] werden dynamisch erzeugt.

Methode: mctBerechnen()

Zur Berechnung der Statistik-Daten für die Statistik-Tabelle STATS_MCT musste die Methode mctBerechnen() implementiert werden, weil die Struktur dieser Tabelle von den anderen Tabellen des Statistiktool-Schemas abweicht und damit die oben genannten Methoden dafür nicht verwendet werden können.

[13] *Abb. 99, S. 129; Abb. 105, S. 133; Abb. 111, S. 137.*

[14] *Abb. 96, S. 127.*

Die Methode erhält das Connection-Objekt und die Variable „bis“ als Parameter. Mit der Variable „bis“ werden die WHERE-Klausel und die SELECT-Anweisung[15] dynamisch erzeugt.

Methode: updateAktStand()

Die Methode updateAktStand() ist die letzte Methode, die im Servlet ServletStatsAktualisieren aufgerufen wird. Diese Methode berechnet keine Statistik-Daten, sondern aktualisiert die Tabelle STATS_AKT_STAND, in der der aktuelle Statistikstand gespeichert ist. Nachdem alle Methoden aufgerufen und alle Statistik-Daten berechnet wurden, muss das Datum der letzten Aktualisierung in die Spalte BIS eingetragen werden, damit diese bei der nächsten Aktualisierung berücksichtigt wird.

Dieser Methode werden ein Connection-Objekt und die Variable „bis“ als Parameter übergeben. Zuerst wird in der Methode überprüft, ob die Variable den Wert 0 hat. Wenn das der Fall ist, wird das Minimum von der Spalte DATUM aus der Tabelle TOOL_AUFGERUFEN ermittelt und in die Spalte VON aus der Tabelle STATS_AKT_STAND gespeichert. Die Spalte VON gibt an, seit wann Daten gesammelt werden. Diese Information wird auf der Benutzeroberfläche des Statistiktools angezeigt. Das gleiche Verfahren muss für das SQL-Abfrageoptimierungstool durchgeführt werden, weil das Tool erst seit 2013 verwendet wird und auch erst seit 2013 Daten dafür gesammelt werden. Wenn der Wert der Variable ungleich 0 ist, dann muss nur die Spalte BIS aktualisiert werden, die auch auf der Benutzeroberfläche des Statistiktools angezeigt wird.

4.2.2.1.2 ServletStatsAlleTools

Das Servlet ServletStatsAlleTools holt die angeforderten Statistik-Daten aus den Statistik-Tabellen STATS_TOOL_AUFGERUFEN und STATS_NATON_AUFGERUFEN, mit denen ein JSON-Objekt erzeugt wird. Anschließend wird das JSON-Objekt an JSP gesendet. Das

[15] *Abb. 81, S. 119.*

JSON-Objekt stellt die Abfrageantwort zur ausgewählten Option von der Gruppe Alle Tools dar, die aus dem Kapitel „3.3.3.1 Informationsstruktur im Statistiktool“ stammt.

Von JSP wird die Abfrage des Benutzers zu diesem Servlet gesendet, die die Option, den Zeitraum und die zum Vergleich ausgewählten Trainer bzw. Tools enthält. Die Option der Abfrage bestimmt, welche SELECT-Anweisungen des Servlets ausgeführt werden sollen. Der Zeitraum der Abfrage wird aus den Werten „von“ und „bis“ gebildet. Der Zeitraum sowie die ausgewählten Trainer werden in die WHERE-Klausel eingefügt und diese in die SELECT-Anweisungen eingebaut.

Das Servlet prüft zuerst, ob das richtige Servlet aufgerufen wurde und ob Statistik-Daten schon vorhanden sind. Danach ermittelt das Servlet, ob der vom Benutzer ausgewählte Zeitraum der Abfrage gültig ist und welche Trainer bzw. Tools vom Benutzer zum Vergleich ausgewählt wurde. Die SELECT-Anweisungen werden entsprechend der Option in einer Switch-Anweisung bestimmt. Nach der Ausführung der SELECT-Anweisungen wird die Ergebnismenge in einem JSON-Objekt gespeichert und an JSP gesendet.

4.2.2.1.3 ServletStatsMCT

Das Servlet ServletStatsMCT ist für das Selektieren der Statistik-Daten aus den Statistik-Tabellen STATS_MCT, STATS_MCT_UNI und STATS_MCT_KATEGORIEN zuständig und verarbeitet die Abfrage nur zum MCT. Mit den Daten wird ein JSON-Objekt erzeugt und an JSP gesendet, das die Abfrageantwort zur ausgewählten Option von der Gruppe MCT in der Informationsstruktur des Statistiktools darstellt.

Nachdem das Servlet die Abfrage von JSP erhalten hat, werden die vom Benutzer ausgewählte Option und die Gültigkeit des Zeitraums, die als Parameter übergeben werden, ausgewertet. Dazu erhält das Servlet ServletStatsMCT den Parameter „hochschule“, in dem die ID der Hochschule steht, die bei dem Selektieren der Daten berücksichtigt werden soll. Wenn der Wert vom Parameter „hochschule“ 0 ist, sollen die Statistik-Daten von allen Hochschulen ohne Trennung selektiert werden. Somit ist die Anforderung zur Datentrennung nach Hochschule vom Kapitel 3.3.3.3 erfüllt.

```
// Falls Daten von nur eine Hochschule ausgewählt werden soll.
String hochschule = request.getParameter("hochschule");
String hochschuleKlausel = null;
// Bei hochschule = 0, Daten von allen Hochschulen laden
if(hochschule.equals("0")) {
      hochschuleKlausel = "";
}
else {
      hochschuleKlausel = "AND (ppf.bereich_id=" + hochschule + ") ";
}
```

Abbildung 124: Trennung nach Hochschule im Servlet ServletStatsMCT

Zusätzlich wird der Parameter „optionAnzahl“ dem Servlet übergeben. Mit diesem Parameter kann die Anzahl von selektierten Datensätzen eingegrenzt werden. Die Eingrenzung ist nur bei SELECT-Anweisungen notwendig, die Datensätze aus Aufgaben des MCTs selektieren, weil bei längerem Zeitraum zu viele Datensätze ausgewählt werden und dadurch die Diagramme unübersichtlich werden. Die Eingrenzung erfolgt in einer äußeren SELECT-Anweisung mit der ROWNUM-Klausel in der WHERE-Klausel, wie der Teil des Quellcodes von der folgenden Abbildung zeigt. Mit ROWNUM kann die Anzahl von Zeilen in einer Ergebnismenge bestimmt werden.

```
// Anzahl der Daten eingrenzen (nur bei Aufgaben)
String eingrenzungKlausel = request.getParameter("optionAnzahl");

            query = "SELECT * " +
                     "FROM ( " +
                       "SELECT TO_CHAR(st.datum, 'DD.MM.YYYY'), st.frage_id, ppf.text,  " +
                                 "st.anzahl_richtig_beantwortet, st.anzahl_aufrufe " +
                      "FROM STATS_MCT st, PLISCHKE_PROJEKT.fragen ppf " +
                      "WHERE (st.datum BETWEEN TO_DATE('" + von + "', 'DD.MM.YYYY') "+
                       "AND TO_DATE('" + bis + "', 'DD.MM.YYYY') )  " +
                      "AND (ppf.frage_id = st.frage_id)  " +
                      "AND (st.anzahl_richtig_beantwortet != 0) " +
                       hochschuleKlausel +
                      "ORDER BY st.datum ASC, st.anzahl_richtig_beantwortet DESC " +
                    ") " +
                   "WHERE ROWNUM <= " + eingrenzungKlausel;
```

Abbildung 125: SELECT-Anweisung im Servlet ServletStatsMCT

4.2.2.1.4 Weitere Servlets

Die Servlets ServletStatsSQLOPT, ServletStatsSQLTrainer, ServletStatsSQLTrainer2 und ServletStatsXQueryTrainer enthalten die gleiche Funktionalitätsstruktur und jeder Trainer wurde zu einem Servlet zugeordnet. Diese Servlets greifen aber auf verschiedene Statistik-Tabellen zu. Das Servlet ServletStatsSQLOPT selektiert Statistik-Daten aus den Tabellen STATS_SQL_OPTIMIZER_SCHEMA und STATS_SQL_OPTIMIZER_FRAGEN. Die Tabellen STATS_SQL_TRAINER_1_SCHEMA und STATS_SQL_TRAINER_1_FRAGEN werden vom Servlet ServletStatsSQLTrainer verwendet. Die Servlets ServletStatsSQLTrainer2 und ServletStatsXQueryTrainer greifen auf die Statistik-Tabellen STATS_SQL_TRAINER_SCHEMA und STATS_SQL_TRAINER_FRAGEN, bzw. STATS_XQUERY_SCHEMA und STATS_XQUERY_FRAGEN zu.

Mit den selektierten Statistik-Daten werden JSON-Objekte erzeugt, die an JSP gesendet werden. Diese Servlets verarbeiten die entsprechenden Optionen der Gruppen SQL-Abfrageoptimierung, SQL-Trainer, SQL-Trainer2 und XQuery, die im Kapitel „3.3.3.1 Informationsstruktur im Statistiktool“ definiert wurden.

Bei allen diesen Servlets werden die vom Benutzer ausgewählte Option und die Gültigkeit des Zeitraums, die als Parameter übergeben werden, ausgewertet. Dazu wird auch der Parameter „optionAnzahl“ übergeben, damit die Anzahl von selektierten Datensätzen eingegrenzt werden kann. Die Eingrenzung wird auch nur bei SELECT-Anweisungen verwendet, die Datensätze aus Aufgaben selektieren. Die Datensätze werden durch eine äußere SELECT-Anweisung mit der ROWNUM-Klausel in der WHERE-Klausel eingegrenzt.

4.2.3 Präsentationsschicht

4.2.3.1 JSP

Die Präsentationsschicht des Statistiktools wird durch nur eine JSP-Datei realisiert. Die JSP-Datei index.jsp stellt die Benutzeroberfläche des Statistiktools dar, die aus HTML-Elementen und Widgets von Dojo besteht.

Bei der Ausführung der JSP-Datei werden zuerst die CSS-Dateien dijit/themes/claro/claro.css und statistiktool.css, die JavaScript-Datei hilfsFunktionen.js und Dojo geladen und eingebunden. Die JavaScript-Datei hilfsFunktionen.js enthält die Funktion alleCheckboxAuswaehlen(), mit der alle Checkbox-Elemente auf der Benutzeroberfläche aus- oder abgewählt werden können, und die Funktion diagAuswaehlen(), mit der die erstellten Diagramme ein- und ausgeblendet werden können. In der CSS-Datei statistiktool.css sind die CSS-Elemente von der Gestaltungsvorlage des EDB-Portals gespeichert, die auf der Benutzeroberfläche des Statistiktools verwendet werden. Diese Elemente wurden aus der Datei style.css entnommen, die auf dem EDB-Portal eingesetzt wird. Die verwendete CSS-Datei dijit/themes/claro/claro.css beinhaltet vorgefertigte CSS-Elemente von Dojo. Einige CSS-Elemente mussten aber verändert werden, damit sie zum Styling des EDB-Portals passen. Die Veränderungen erfolgten, indem der Inhalt von einigen CSS-Elementen auskommentiert bzw. mit anderen Werten ersetzt wurde, wie die folgende Abbildung zeigt.

```
.claro .dijitDialog {
        /* border: 1px solid #759dc0; */
        border: 1px solid #0F3764;
        /*
        -webkit-box-shadow: 0 1px 5px rgba(0, 0, 0, 0.25);
        -moz-box-shadow: 0 1px 5px rgba(0, 0, 0, 0.25);
        box-shadow: 0 1px 5px rgba(0, 0, 0, 0.25);
        */
}
.claro .dijitDialogTitle {
        padding: 0 1px;
        font-size: 1.091em;
        /*color: #000000;*/
        color: #FFFFFF;
        /*font-weight: bold;*/
}
```

Abbildung 126: Veränderungen in der Datei dijit/themes/claro/claro.css

Zur Darstellung von einigen Daten auf der Benutzeroberfläche müssen diese von Tabellen des Statistiktool-Schemas ausgelesen werden. Das Statistiktool enthält eine Anzeige mit dem Zeitraum, in dem Statistik-Daten zur Abfrage des Benutzers zur Verfügung stehen. Diese Daten werden von Spalten VON und BIS aus der Tabellen STATS_AKT_STAND geladen. Auf der Anzeige steht beispielweise der Text „Aktueller Statistikstand: 02.04.2012 – 12.08.2013“.

Neben dem Text wurde ein Button mit dem Label „Statistik aktualisieren“ platziert, der den Servlet ServletStatsAktualisieren aufruft, damit dieses die Statistik-Daten berechnet und in die Statistik-Tabellen speichert. Wenn noch keine Statistik-Daten berechnet wurden, steht in den Spalten der Wert NULL und auf der Anzeige wird der Text „Es ist noch keine Statistik vorhanden“ ausgegeben. Ohne dass die Berechnung der Statistik-Daten schon einmal durchgeführt wurde, ist die Ausführung einer Abfrage noch nicht möglich, und eine Fehlermeldung wird ausgegeben. Dazu werden auch die Daten der Hochschulen aus der Tabelle BEREICHE des Schemas Plischke_Projekt ausgelesen, damit diese für die Abfrage im MCT zur Auswahl stehen.

Die Abfragen im Statistiktool werden durch HTML-Formulare dargestellt. Diese Formulare bestehen aus mehreren Elementen, mit denen die Abfragen eingestellt werden. Die Formulare wurden auf Basis der gebildeten Gruppen aus dem Kapitel „3.3.3.1 Informationsstruktur im Statistiktool“ aufgeteilt. Somit gibt es sechs Formulare, die den Gruppen Alle Tools, MCT, SQL-Trainer, SQL-Trainer2, XQuery und SQL-Abfrageoptimierungstool entsprechen. Die Formulare sind in Registerkarten eingeordnet und können dort ausgewählt werden.

Alle Formulare enthalten bestimmte Elemente, wie beispielsweise ein Option-Element, das aus der entsprechenden Gruppe stammt und zeigt, welche Daten abgefragt werden sollen. Zu den Elementen der Formulare gehören auch ein Von- und Bis-Element, die den möglichen Zeitraum für die Abfrage darstellen. Diese Elemente bestehen aus einem Tag-Element, Monat-Element und Jahr-Element, die einzeln eingestellt werden können. Die Formulare haben auch ein Eingrenzen-Element, das die Anzahl von Datensätze bei Abfragen über Aufgaben der Trainer eingrenzt. Im Eingrenzen-Element können die Werte 5, 10, 15, 25, 50 oder 100 ausgewählt werden. Dazu gehört auch ein Ausführungsbutton, um die Abfrage zu starten, indem die Daten im Formular an das entsprechende Servlet gesendet werden.

Im Formular zur Gruppe Alle Tools können die Trainer angekreuzt werden, die in der Abfrage verglichen werden sollen. Diese werden als Checkboxen dargestellt. Im Formular zur Gruppe MCT gibt es zusätzlich die Möglichkeit, eine bestimmte Hochschule auszuwählen, damit nur Statistik-Daten aus dieser Hochschule in der Abfrage berücksichtigt werden.

Alle Servlets warten auf eine Anfrage von einem Client, die von JSP weitergeleitet wird. Die Anfrage eines Client erfolgt erst nach der Einstellung der Anfrage im Formular und dem Anklicken des Ausführungsbuttons, das das Ereignis in JSP startet.

Zu jedem Formular gehört ein Dojo-Modul in JSP, das die Anfrage vom Client entgegennimmt und an das Servlet mittels AJAX-Kommunikation weiterleitet. Jedes Modul enthält eine Dojo-Funktion on(), die die gestarteten Ereignisse behandelt. In dieser Funktion gibt es die Funktion request.post(), die die Daten des Formulars an ein bestimmtes Servlet sendet. Das Servlet sendet dann ein JSON-Objekt mit den Statistik-Daten als Antwort zurück.

In JSP wird das Element „fehlermeldung“ vom JSON-Objekt überprüft. Wenn das Element leer ist, ist kein Fehler im Servlet aufgetreten. Somit können die Funktionen der Module aufgerufen werden, die die Diagramme im Statistiktool erstellen. Das JSON-Objekt wird den Funktionen übergeben. Wenn das Element eine Fehlermeldung enthält, wird ein Dialog vom Paket Dijit erzeugt und angezeigt. Die entsprechende Fehlermeldung wird auf dem Dialog ausgegeben. Fehler, die in der Übertragung von Daten auftreten, werden auch behandelt, indem die entsprechende Fehlermeldung auf einem Dialog angezeigt wird.

In der folgenden Abbildung wird das Modul zur AJAX-Kommunikation für das Formular von MCT.

```
<!-----------------    AJAX-Kommunikation für MCT   ----------------------->
<script>
        require(["dojo/dom", "dojo/on", "dojo/request", "dojo/dom-form",
                "statsDiagramme/kreisDiagramm", "statsDiagramme/stabDiagramm",
                "dojo/json", "dojox/json/query", "dijit/Dialog", "dijit/form/Button", "dojo/domReady!"],
        function(dom, on, request, domForm, kreisdiagramm, stabdiagramm, json){
                var form = dom.byId('mctForm');    // Legt fest, welche Formular behandelt wird.
                on(form, "submit", function(evt){  // Ereignishandler bei Submit des Formulars

                        evt.stopPropagation();      // Verhindert Weitergabe vom Ereignis.
                        evt.preventDefault();// Führt return false von HTML-Element durch.

                        request.post("ServletStatsMCT", {            // Daten an den Servlet abgesendet
                                data: domForm.toObject("mctForm"),         // Daten des Formulars
                                handleAs: "json" // Bearbeitung von JSON-Objekt ermöglichen

                        }).then(function(response){
                                var fehler = dojox.json.query("fehlermeldung", response);
                                if(fehler == ""){
                                        // Module mit Daten aufrufen, um die Diagramme zu erstellen.
                                        stabdiagramm.setStabDiagramm(response);
                                        kreisdiagramm.setKreisDiagramm(response);
                                        dom.byId("statsMenuButton").style.visibility = 'visible';
                                }
                                else {
                                        // Fehlermeldung ausgegeben (interner Fehler)
                                        var dialog5 = new dijit.Dialog({
                                                title: "Fehler",
                                                style: "width:500px;",
                                                content: fehler + "<p /><div "+
                "class=\"buttonSchliessen\"><button data-dojo-type=\"dijit/form/Button\" ty-
        pe=\"submit\">Schlie&szlig;en</button></div>"
                                        });
                                        dialog5.show();
                                }
                        }, function(error) {
                                // Dialogfenster erstellen und Fehlermeldung ausgegeben (externer Fehler)
                                var dialog6 = new dijit.Dialog({
                                        title: "Fehler",
                                        style: "width:500px;",
                                        content: error + "<p /><div class=\"buttonSchliessen\">"+
                                                "<button data-dojo-type=\"dijit/form/Button\" "+
                                                "type=\"submit\">Schlie&szlig;en</button></div>"
                                });
                                dialog6.show();
                        });
                        });
                }
        );
</script>
```

Abbildung 127: Modul im JSP zur AJAX-Kommunikation

4.2.3.2 Module zur Erstellung der Diagramme

Für das Statistiktool wurden zwei Module in JavaScript implementiert, die Diagramme erstellen und diese auf die Benutzeroberfläche anzeigen. Diese Module befinden sich im Paket dojo/statsDiagramme und werden bei der Ausführung der JSP-Datei index.jsp von Dojo-Loader mitgeladen.

```
<script src="dojo-1.9.1/dojo/dojo.js"
                data-dojo-config="async: true,
                                  packages: [{
                                      name: 'statsDiagramme',
                                      location: 'statsDiagramme'
                                  }]"></script>
```

Abbildung 128: Laden des Pakets statsDiagramme

Die Implementierung der Diagramme in Modulen vereinfacht die Erweiterung des Statistiktools, indem neue Module einfach in das Paket dojo/statsDiagramme kopiert und in der JSP-Datei aufgerufen werden müssen.

Das Modul dojo/statsDiagramme/kreisDiagramm enthält die Funktion setKreisDiagramm(), mit der Kreisdiagramme erstellt werden können. Im Modul dojo/statsDiagramme/stabDiagramm steht die Funktion setStabDiagramm(), die Stabdiagramme erstellt.

Diese Funktionen werden in JSP erst aufgerufen, nachdem das JSON-Objekt vom Servlet angekommen ist und überprüft wurde, dass kein Fehler im Servlet aufgetreten ist.

Der Ablauf der Funktionen in den Modulen entspricht dem Ablauf aus dem Kapitel „3.4.4 Erstellung eines Diagramms mit DOJO“. In beiden Modulen werden auch zuerst die folgenden Dojo-Module geladen: dojox/charting/Chart, dojox/charting/themes/Claro, dojox/charting/plot2d/Markers und Dojox/charting/axis2d/Default. Im Modul dojo/statsDiagramme/kreisDiagramm wird auch das Modul dojox/charting/plot2d/Pie geladen, das das Kreisdiagramm darstellt. Das Modul dojo/statsDiagramme/stabDiagramm benötigt

das Modul dojox/charting/plot2d/Columns, um das Stabdiagramm zu erstellen. Sowohl die Kreissektoren des Kreisdiagramms als auch die Stäbe des Stabdiagramms werden entsprechend der Daten aus dem JSON-Objekt, die in ein Array gespeichert werden, automatisch erzeugt.
In beiden Modulen wird auch das Modul dojox/charting/action2d/Tooltip geladen, das ein kleines Pop-up-Fenster anzeigt, wenn der Benutzer den Mauszeiger über die Kreissektoren und Stäbe bewegt. Dazu wird im Modul dojo/statsDiagramme/kreisDiagramm das Modul dojox/charting/action2d/MoveSlice verwendet, damit sich die Kreissektoren bei Mousehover-Ereignis leicht vergrößern, um die Sichtbarkeit von kleinen Kreissektoren zu verbessern.

Nachdem alle benötigten Module geladen wurden, werden die Statistik-Daten aus dem JSON-Objekt response in ein Array gespeichert, wie die folgende Abbildung zeigt.

```
// Erzeugung des Array mit dem Schlüssel summe von JSON-Objekt im summeArray
var daten = new Array(response.summeArray.length);
// Array mit dem Daten aus dem JSON-Objekt
for(var i=0; i < response.summeArray.length; i++) {
        daten[i] = { x: 1, y: response.summeArray[i].summe };
}
```

Abbildung 129: Statistik-Daten in ein Array speichern

Danach wird das Diagramm-Objekt erzeugt und ein Thema zum Diagramm hinzugefügt. Mit der Methode addPlot wird der Typ des Diagramms definiert und es ist auch möglich, weitere Konfigurationen durchzuführen. Mit addSeries werden die Daten des Arrays in das Diagramm eingefügt. Dazu wird das Diagramm mit den Modulen Tooltip und MoveSlice erweitern. Anschließend wird das Diagramm gerendert. Die folgende Abbildung zeigt diesen Ablauf.

```
var kreisDiag = new Chart("statsKreisDiagramm");
kreisDiag.setTheme(theme);
kreisDiag.addPlot("default", {
        type: PiePlot,
        radius: 180,
        fontColor: "black",
        labelOffset: 50
        });
kreisDiag.addSeries("Zusammenfassung", daten);
var tip = new Tooltip(kreisDiag, "default");
var mag = new MoveSlice(kreisDiag,"default");
kreisDiag.render();
```

Abbildung 130: Erstellung eines Kreisdiagramms

Auf der Benutzeroberfläche werden die Diagramme in Div-Tags „statsKreisDiagramm“ und „statsStabDiagramm“ der JSP-Datei dargestellt. Es wird aber immer einzeln als Diagramm angezeigt. Welches Diagramm angezeigt werden soll, kann der Benutzer auswählen, indem er den Button des Menüs anklickt, das sich unterhalb der Diagramme befindet. Mit Anklicken des Buttons wird die JavaScript-Funktion diagAuswaehlen() aufgerufen, die die Diagramme aus- und einblendet.

4.2.3.3 Gestaltung der Benutzeroberfläche

Die Benutzeroberfläche des Statistiktools wurde mit vorgefertigten Widgets des Dijit-Pakets gestaltet, die die Komponenten der Benutzeroberfläche darstellen. Ein BordContainer bildet das Grundgerüst der Oberfläche ab, der in fünf Regionen aufgeteilt werden kann. Für die Benutzeroberfläche werden nur die Regionen Top und Center verwendet. Die Top-Region besteht aus einem ContentPane, auf dem der Head-Teil des Statistiktools platziert wird. In der Center-Region befindet sich ein TabContainer, der den Body-Teil des Statistiktools darstellt. Der TabContainer enthält mehrere ContentPanes, auf denen die Formulare und Diagramme angezeigt werden.

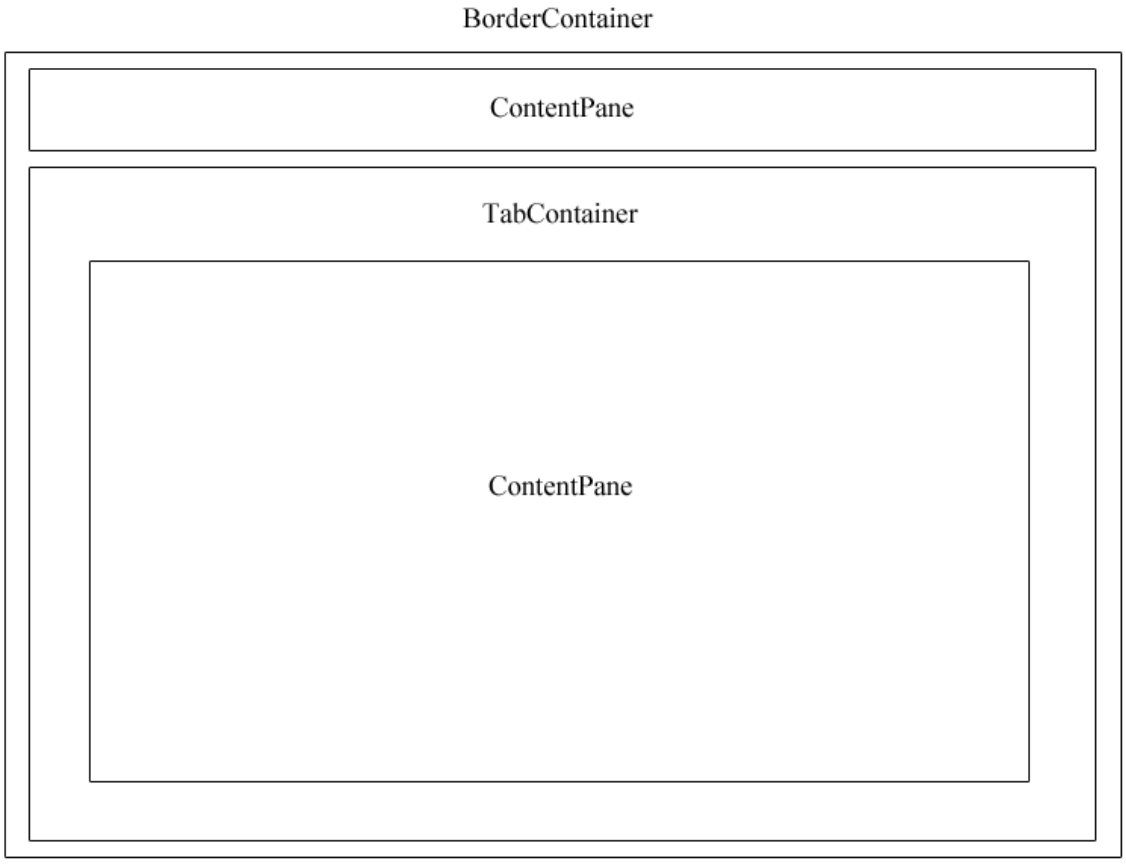

Abbildung 131: Verwendete Widgets in der Benutzeroberfläche

Im Head-Teil des Statistiktools kommen das Logo des EDB-Portals und der Titel vor. Das Logo ist auch ein Link zum EDB-Portal. Auf der rechten Seite des Head-Teils befindet sich der Statistik-Aktualisieren-Button, mit dem der Servlet ServletStatsAktualisieren zur Berechnung der Statistik-Daten aufgerufen werden kann.

Im Body-Teil des Statistiktools stehen die Formulare, die in Form von Registerkarten dargestellt werden.

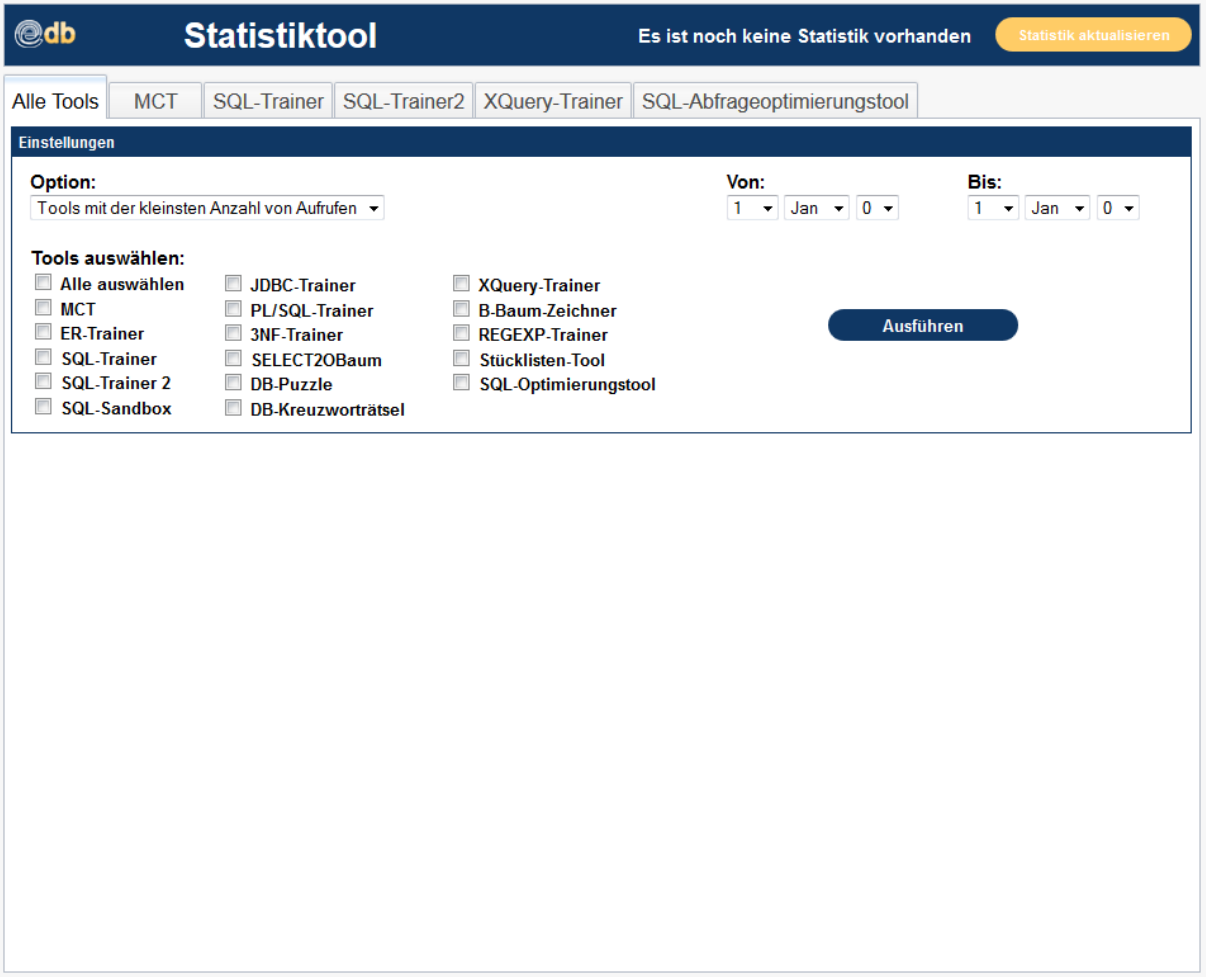

Abbildung 132: Benutzeroberfläche des Statistiktools

Wenn noch keine Statistik vorhanden ist, kann noch keine Abfrage ausgeführt werden. Zuerst muss der Statistik-Aktualisieren-Button angeklickt werden, damit die Statistik-Aktualisierung durchgeführt wird. Während der Durchführung der Funktion wird ein Dialog angezeigt, damit der Benutzer wartet, bis der Vorgang abgeschlossen ist.

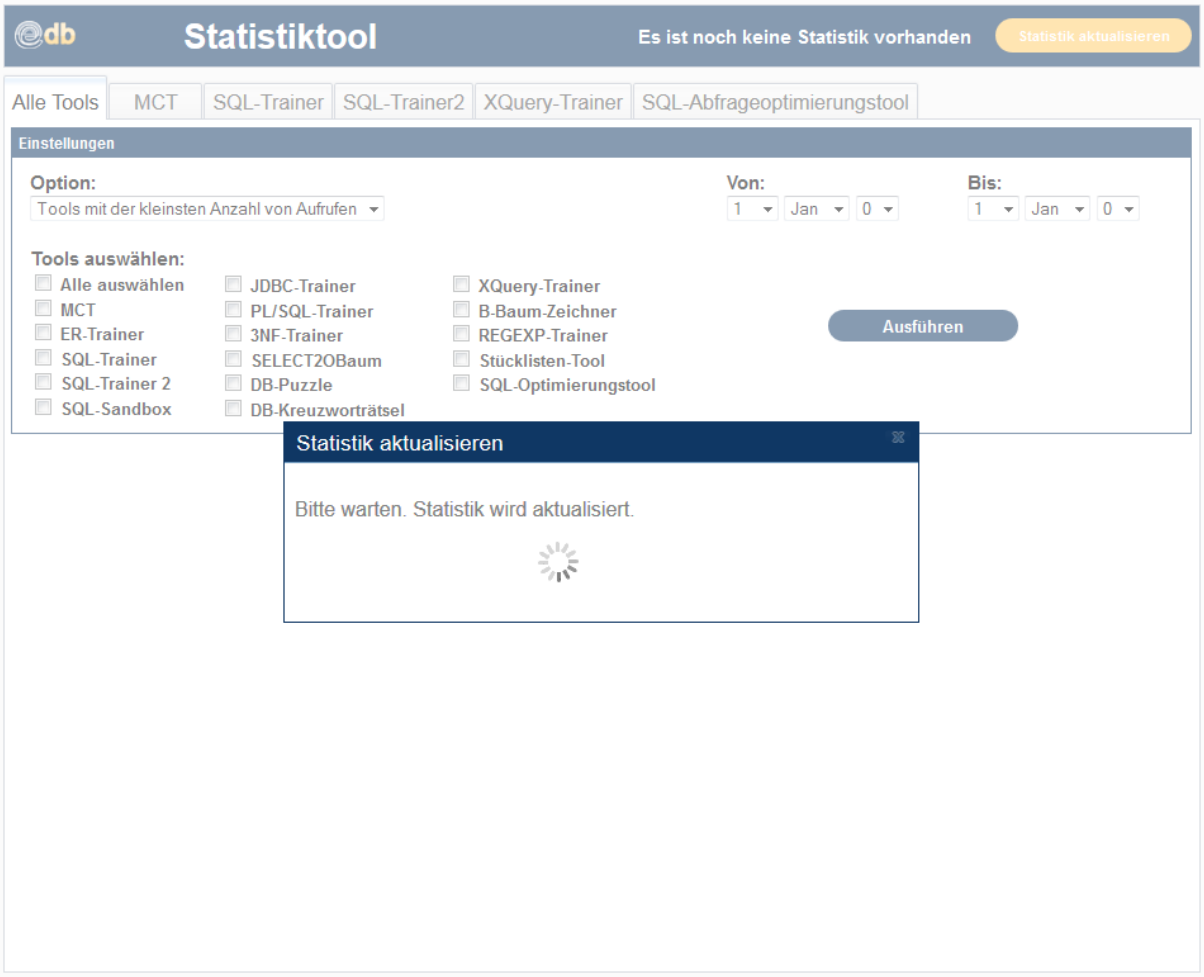

Abbildung 133: Dialog bei der Aktualisierung der Statistik-Daten

Nach der Durchführung der Statistik-Aktualisierung wird der Dialog geschlossen und das Statistiktool wird neu geladen, damit der aktuelle Statistikstand angezeigt wird.

Der Benutzer kann dann mehrere Einstellungen in einem Formular vornehmen, bevor der Ausführungsbutton betätigt wird, um eine Abfrage auszuführen. Dabei wird ein entsprechender Servlet aufgerufen.

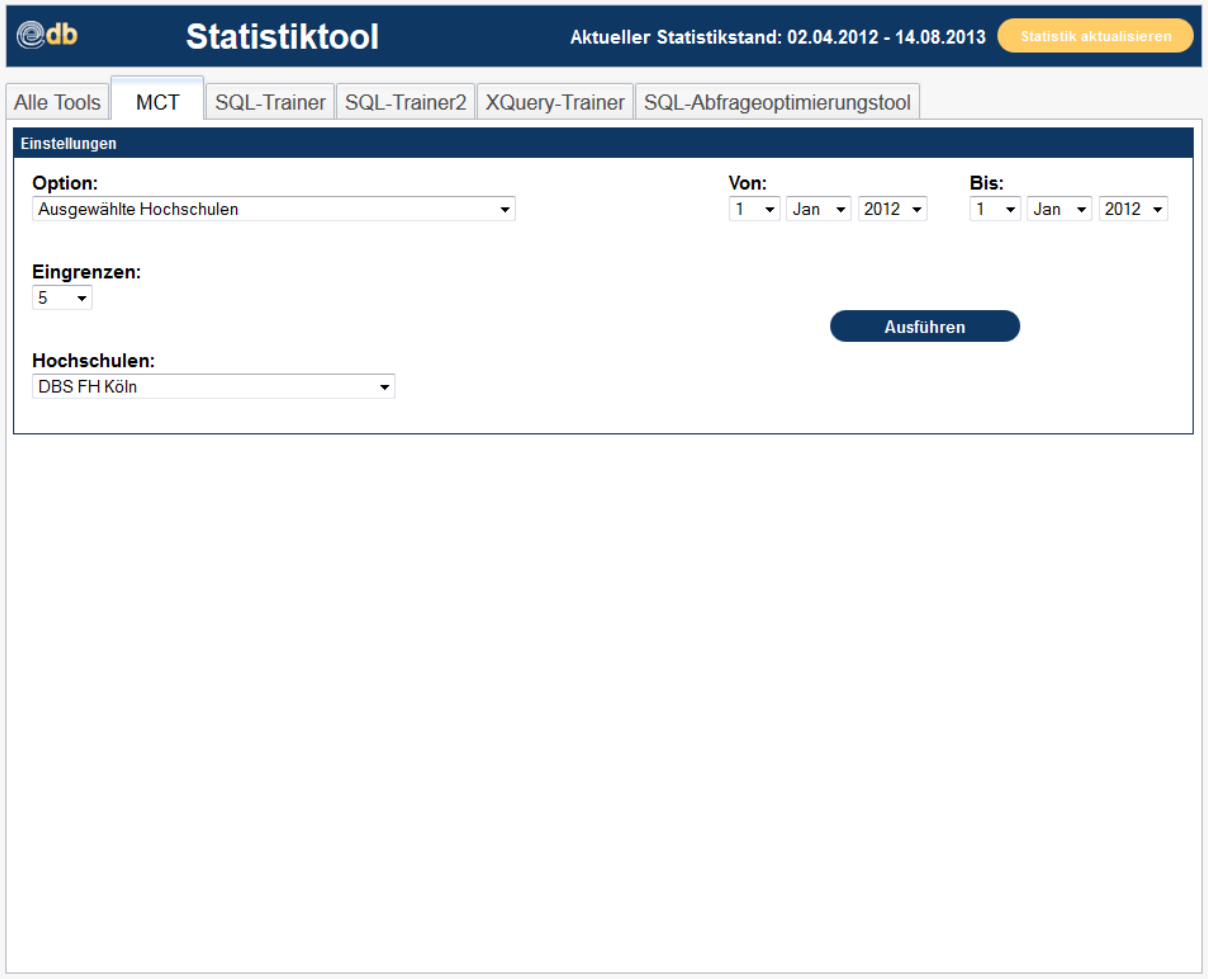

Abbildung 134: Formular vom MCT auf der Benutzeroberfläche

Falls ein Fehler im Statistiktool auftritt, oder der Benutzer eine falsche Eingabe macht, wird ein Dialog mit der entsprechenden Fehlermeldung angezeigt.

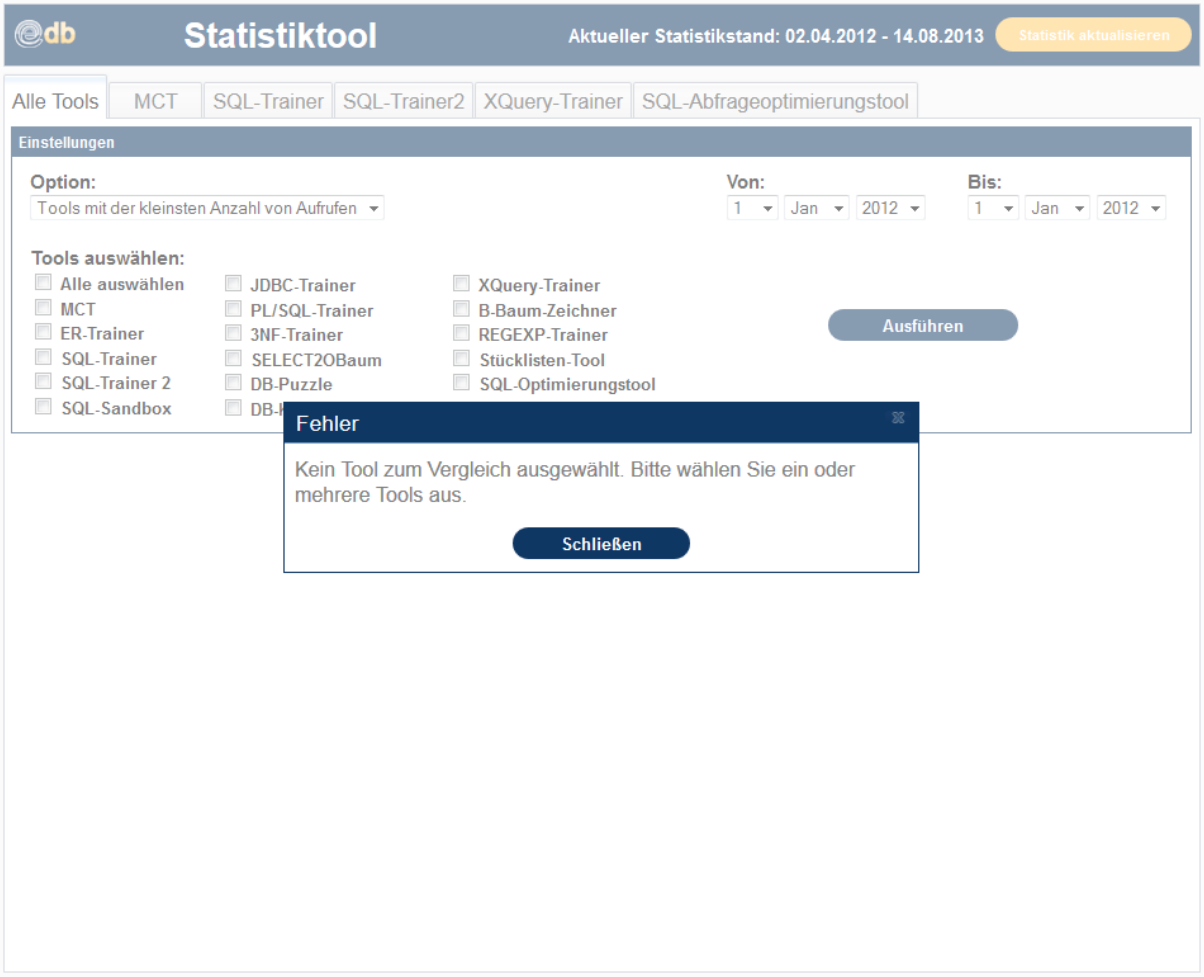

Abbildung 135: Anzeige einer Fehlermeldung im Statistiktool

Falls die Abfrage erfolgreich ausgeführt wurde, wird zuerst das Kreisdiagramm als Ergebnis der Abfrage unterhalb des Formulars angezeigt. Unter dem Diagramm wird auch ein Menü dargestellt, das zwei Buttons enthält. Mit diesen Buttons kann der Benutzer zwischen dem Kreisdiagramm und dem Stabdiagramm umschalten.

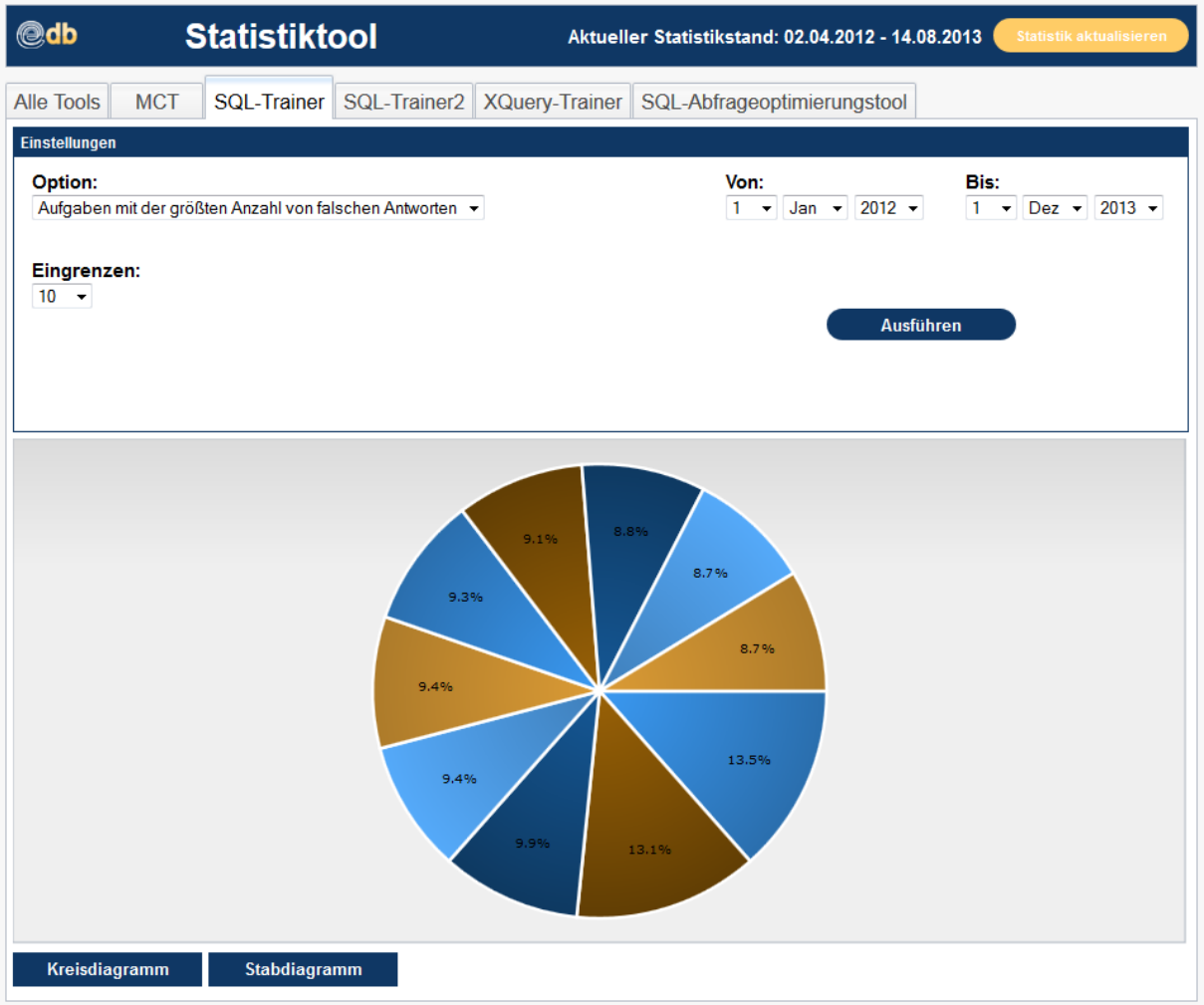

Abbildung 136: Kreisdiagramm im Statistiktool

Wenn der Benutzer den Stabdiagramm-Button anklickt, wird das Stabdiagramm auch als Ergebnis der Abfrage eingeblendet und das Kreisdiagramm ausgeblendet. Der Benutzer kann immer wieder eine neue Abfrage starten.

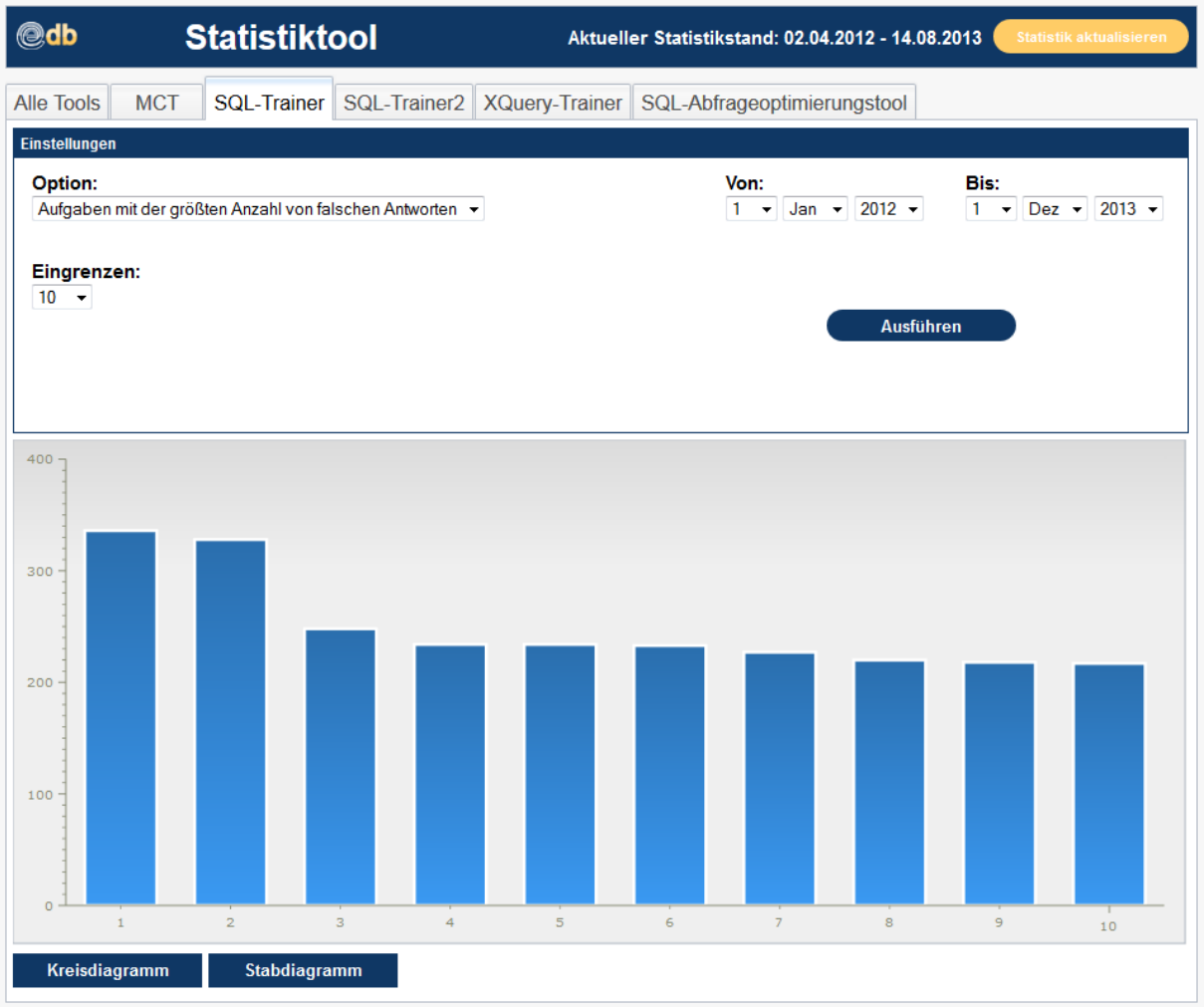

Abbildung 137: Stabdiagramm im Statistiktool

5. Fazit

In diesem Kapitel werden die gewonnenen Erkenntnisse aus dieser Diplomarbeit zusammengefasst und bewertet.

Der erste Teil der Arbeit bestand darin, die Konzeption des Statistiktools zu entwickeln, in der die Analyse des Statistik-Schemas im Vordergrund stand. Die gründliche Analyse hat sich als sehr nützlich erwiesen. Mit den Erkenntnissen aus dieser Analyse konnten die Struktur und die Eigenschaften der Tabellen bestimmt werden. Dazu wurden die Daten der Tabellen ausgewertet, mit denen die Statistik erstellt werden sollen. Die Erkenntnisse zeigten aber auch, dass die Tabellen sehr groß sind und keine Indizes haben. Dadurch konnte ein Performance-Problem erkannt werden, das durch die Überprüfung von Ausführungsplänen bestätigt wurde.

In dem zweiten Teil der Realisierung wurde das Statistiktool implementiert. Daraus folgten zuerst die Erstellung der Statistik-Tabellen und die Berechnung der Statistik-Daten als Lösung, um das Performance-Problem zu vermeiden. Die Implementierung des Statistiktools als JSP-Anwendung mit der Verbindung zu Servlets (Java-EE-Technologie) war von Vorteil, weil diese sich schon mehrmals in der Praxis bewährt hat. Zur Übertragung von Daten wurde das Datenformat JSON verwendet, mit dem Daten schneller als mit XML ausgetauscht werden können. Die Auswahl und die Verwendung von Dojo Toolkit im Statistiktool haben sich als sehr sinnvoll erwiesen. Die Implementierung von Diagrammen und der Benutzeroberfläche des Statistiktools konnte mit den vorgefertigten Widgets und Modulen vereinfacht werden. Dazu konnten die Anpassungen und das Styling der Widgets mit dem CSS-Thema und dem Hover-Effekt auch ohne Einschränkungen eingesetzt werden. Das Paketsystem und die Namenräume von Dojo ermöglichen auch, das Statistiktool mit neuen Modulen einfach zu erweitern. So konnten alle Anforderungen an das Statistiktool erfüllt werden.

Mit dem Statistiktool konnte ein Tool realisiert werden, das sehr einfach zu verwenden ist und trotz großer Menge an Daten eine gute Performance aufweist. Die Benutzeroberfläche des Statistiktools ermöglicht eine übersichtliche Ansicht der Ergebnisdaten, damit diese durch den Benutzer einfach und schnell erfasst werden können.

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AMD	Asynchronous Module Definition
APEX	Oracle Application Express
AJAX	Asynchronous JavaScript and XML
CSS	Cascading Style Sheets
DOM	Document Object Model
EDB-Portal	e-Learning-Datenbank-Portal
EJB	Enterprise JavaBean
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ID	Identifier
Java-EE	Java Plattform-Enterprise Edition
JDBC	Java Database Connectivity
JSON	JavaScript Object Notation
JSP	Java Server Pages
MCT	Multiple Choice Test
MCV	Modell-View-Controller
SQL	Structured Query Language
XML	Extensible Markup Language
WAR-Datei	Web-Archiv-Datei
WYSIWYG	What You See Is What You Get

Literaturverzeichnis

Andrew (2008):
Andrew, R.: Der CSS-Problemlöser - Über 100 Lösungen für Cascading Stylesheets. 2. Aufl., Verlag: dpunkt.verlag, Heidelberg 2008.

Evans (2013):
Evans, I.: The Java EE 7 Tutorial: Java EE 7 APIs, http://docs.oracle.com/javaee/7/tutorial/doc/overview008.htm, Download am 30.06.2013 um 22:30 Uhr

Evans/Jendrock (2013):
Evans, I/Jendrock, E.: The Java EE 7 Tutorial: Java Servlet technology, http://docs.oracle.com/javaee/7/tutorial/doc/servlets.htm#BNAFD, Download am 06.07.2013 um 15:10 Uhr

Evans/Jendrock (2013):
Evans, I/Jendrock, E.: The Java EE 7 Tutorial: JSON Processing, http://docs.oracle.com/javaee/7/tutorial/doc/jsonp.htm#GLRBB, Download am 06.07.2013 um 15:30 Uhr

Evans/Jendrock (2013):
Evans, I/Jendrock, E.: The Java EE 7 Tutorial: Java EE Servers, http://docs.oracle.com/javaee/7/firstcup/doc/java-ee003.htm#GCRKQ, Download am 30.06.2013 um 23:20 Uhr

Faeskorn-Woyke/Bertelsmeier (2007):
Faeskorn-Woyke, H./Bertelsmeier, B./ et al.: Datenbanksysteme - Theorie und Praxis mit SQL2003, Oracle und MySQL. 1. Aufl., Verlag: Pearson Studium, München 2007.

Forster (o. J.):
Forster, S.: The Dojo Toolkit - Configuring Dojo with dojoConfig, http://dojotoolkit.org/documentation/tutorials/1.9/dojo_config/, Download am 24.07.2013 um 22:00 Uhr

Forster (o. J.):
Forster, S.: The Dojo Toolkit – Layout with Dijit, http://dojotoolkit.org/documentation/tutorials/1.9/dijit_layout/, Download am 24.07.2013 um 23:30 Uhr

Frischalowski/Böttcher (2007):
Frischalowski, D./Böttcher, U.: Java 6 - Einstieg und professioneller Einsatz. 1. Aufl., Verlag: entwickler.press, Siegen 2007.

Gamperl (2007):
Gamperl, J.: AJAX - Grundlagen, Frameworks, APIs. 2. Aufl., Verlag: Galileo Press, Bonn 2007.

Gamrat (o. J.):
Gamrat, B.: The Dojo Toolkit – Ajax with dojo/request, http://dojotoolkit.org/documentation/tutorials/1.9/ajax/, Download am 24.07.2013 um 23:00 Uhr

Harris (2009):
Harris, R.: Murach's JavaScript and DOM Scripting. 1. Aufl., Verlag: Mike Murach & Associates Inc., USA 2009.

Mintert/Leisegang (2007):
Mintert, S./Leisegang, C.: AJAX - Grundlagen, Frameworks und Praxislösungen. 1. Aufl., Verlag: dpunkt.verlag, Heidelberg 2007.

Nix (2007):

Nix, M./et al.: Exploring JavaScript - OOP, Ajax und Web 2.0. 1. Aufl., Verlag: entwickler.press, Siegen 2007.

Olson (2007):

Olson S.: Ajax und Java. 1. Aufl., Verlag: O'Reillys, Köln 2007.

o. V.: (o. J.)

o. V.: Learn more about Oracle Application Express

http://apex.oracle.com/pls/otn/f?p=4600:6:0::NO:::, Download am 10.08.2013, um 17:00 Uhr.

o. V. (2002)

o. V.: Einführung in JSON
http://www.json.org/json-de.html, Download am 27.07.2013, um 13:45 Uhr.

o. V. (2007):

o. V.: Servlet,

http://faculty.inverhills.mnscu.edu/speng/cs1127/Notes/Ch26/Servlet.jpg, Download am 27.06.2013 um 17:10 Uhr

o. V. (2010):

o. V.: Servlet Overview,

http://docs.oracle.com/cd/B14099_19/web.1012/b14017/overview.htm, Download am 27.06.2013 um 17:00 Uhr.

o.V. (2012):

o.V.: Dojo Toolkit Reference Guide – The Dojo Toolkit – Reference Guide,

http://dojotoolkit.org/reference-guide/1.9/, Download am 21.07.2013 um 13:00 Uhr

o.V. (2013):

o.V.: Servlet – Wikipedia,

http://de.wikipedia.org/wiki/Servlet, Download am 29.06.2013 um 13:45 Uhr

Rittmeyer (2007):
Rittmeyer, W.: JSP-Tutorial,
http://www.jsptutorial.org/content/architecture, Download am 02.07.2013 um21:30 Uhr

Robbins (2006):
Robbins, J. N.: HTML & XHTML - Kurz und gut. 3. Aufl., Verlag: O'Reillys, Köln 2006.

Russel (2008):
Russel, M.: DOJO - The Definitive Guide. 3. Aufl., Verlag: O'Reillys, Sebastopol (USA) 2008.

Seeboerger-Weichselbaum (2007):
Seeboerger-Weichselbaum, M.: AJAX GE-PACKT. 1. Aufl., Verlag: mitp, Heidelberg 2007.

Seeboerger-Weichselbaum (2007):
Seeboerger-Weichselbaum, M.: JavaScript. 4. Aufl., Verlag: bhv, Heidelberg 2007.

Schiefer (o. J.):
Schiefer, B.: Entwicklung von Web-Anwendungen auf Java EE Basis,
http://www.fh-kl.de/~schiefer/lectures/java_ee.php, Download am 01.07.2013 um 16:00 Uhr

Singh (o. J.):
Singh, M.: The Dojo Toolkit - Introduction to AMD Modules,
http://dojotoolkit.org/documentation/tutorials/1.9/modules/, Download am 24.07.2013 um 22:00 Uhr

Snover (o. J.):
Snover, C.: The Dojo Toolkit – Hello Dojo!,
http://dojotoolkit.org/documentation/tutorials/1.9/hello_dojo/, Download am 20.07.2013 um 15:00 Uhr

Steyer (2003):
Steyer, R.: JavaScript in 21 Tagen. 1. Aufl., Verlag: Markt+Technik Verlag, München 2003.

Steyer (2008):
Steyer, R.: AJAX Frameworks - RIAs mit DOJO & Co. 1. Aufl., Verlag: Addison-Wesley Verlag, München 2008.

Turau (2001):
Turau, V.: Java Server Pages - Dynamische Generierung von Web-Dokumenten. 2. Aufl., Verlag: dpunkt.verlag, Heidelberg 2001.

Turau/Saleck/Lenz (2004)
Turau, V./Saleck, K./Lenz, C.: Web-basierte Anwendungen entwickeln mit JSP2. 2. Aufl., Verlag: dpunkt.verlag, Heidelberg 2004.

Ullenboom (2005):
Ullenboom, C.: Java ist auch eine Insel – 17 Servlets und Java Server Pages, http://openbook.galileodesign.de/javainsel5/javainsel17_000.htm, Galileo Press GmbH, Download am 02.07.2013 um 22:00 Uhr

Walsh (o. J.):
Walsh, D.: The Dojo Toolkit – Dojo Charting, http://dojotoolkit.org/documentation/tutorials/1.9/charting/, Download am 25.07.2013 um 13:00 Uhr

Walsh (o. J.):
Walsh, D.: The Dojo Toolkit – Advanced Charting with Dojo, http://dojotoolkit.org/documentation/tutorials/1.9/charting_advanced/, Download am 25.07.2013 um 14:00 Uhr

Wutka (2002)
Wutka, M.: J2EE Developer's Guide - JSP, Servlets, EJB 2.0, JNDI, JMS, JDBC, CORBA, XML, RMI. 1. Aufl., Verlag: Markt+Technik Verlag, München 2002.

Zakas/McPeak/Fawcett (2006):
Zakas, N./McPeak, J./Fawcett, J.: Ajax Professionell. 1. Aufl., Verlag: mitp, Heidelberg 2006.

Anhang

Anhänge befinden sich in elektronischer Form auf der beiliegenden CD.

- Diplomarbeit als PDF-Datei
- Eclipse-Projekt
- SQL-Skripte
- WAR-Datei

Erklärung über die selbständige Abfassung der Arbeit

Ich versichere, die von mir vorgelegte Arbeit selbständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Gummersbach, den 29.08.2013 _____

(Eduardo Wildt Graziani)